Vol.9 No.2 June 1990

BEEBUG

FOR THE BBC MICRO & MASTER SERIES

Spline Text

- A VERSATILE CHARACTER EDITOR
- TURMITES
- DECIMAL SQUEEZE
- FOG INDEX

BEEBUG Vol.9 No.2 June 1990

FEATURES

SplineText 6
Turmites 11
Storing Variables and Procedures in SWR on a Model B 15
Fog Index 19
A Versatile Character Editor 22
Practical Assembler (Part 2) 27
Decimal Squeeze 32
First Course - A Menu Routine 36
Designer Shoot-'Em-Up (Part 2) 39
512 Forum 43
Wordwise Plus Auto-Backup Utility 46
Music Programming In Ample (Part 3) 48

REVIEWS

Apricote Studios' Personal Accounts 30
Adventure Games 53

REGULAR ITEMS

Editor's Jottings 4
News 5
Points Arising 42
Bulletin Boards 54
Hints and Tips 57
RISC User 56
Best of BEEBUG 58
Postbag 59
Personal Ads 60
Subscriptions & Back Issues 62
Magazine Disc/Cassette 63

HINTS & TIPS

Z88 to BBC Micro
BEEBUG Toolkit on the Master
Reconfiguring the Master

PROGRAM INFORMATION

All listings published in BEEBUG magazine are produced directly from working programs. They are formatted using LISTO 1 and WIDTH 40. The space following the line number is to aid readability only, and may be omitted when the program is typed in. However, the rest of each line should be entered exactly as printed, and checked carefully. When entering a listing, pay special attention to the difference between the digit one and a lower case l (L). Also note that the vertical bar character (Shift \) is reproduced in listings as |.

All programs in BEEBUG magazine will run on any BBC micro with Basic II or later, unless otherwise indicated. Members with Basic I are referred to the article on page 44 of BEEBUG Vol.7 No.2 (reprints

available on receipt of an A5 SAE), and are strongly advised to upgrade to Basic II. Any second processor fitted to the computer should be turned off before the programs are run.

Where a program requires a certain configuration, this is indicated by symbols at the beginning of the article (as shown opposite). Any other requirements are referred to explicitly in the text of the article.

Program will not function on a cassette-based system.

Program needs at least one bank of sideways RAM.

Program is for Master 128 and Compact only.

Editor's Jottings

THE BEEBUG RETAIL CATALOGUE

It has been our regular practice over many years now to mail out a Retail Catalogue with each issue of BEEBUG magazine. As readers will be aware, the catalogue has considerably increased in size over the last 12 months, culminating in the 72 page summer edition which you should have received with the May issue of BEEBUG. Because of this, we will no longer publish and send out a new version of the catalogue with every issue. Instead, the full catalogue will be sent out three or four times a year, with short supplements in between where this is appropriate.

So before you lift the phone to find out where your latest catalogue is, please check to locate last month's issue, and keep that for reference until you receive the next catalogue, due in September. If you have lost or mislaid the summer edition we can supply a further copy on request. We hope that in this way we can provide a more comprehensive mail-order service, and improve on the competitiveness of our prices.

EDIKIT

I would like to remind all readers of the new EdiKit ROM which we launched last month. Based on the successful series of articles by Bill Hine, and incorporating all the routines from our previous Basic Booster, this provides an excellent toolkit of functions and routines which will be of help to all Basic programmers, and at a special low price for BEEBUG members. Furthermore, the coding has been tested and suitably modified so that EdiKit will function correctly on the model B, The Master 128 and Master Compact, and with the latest Master ROM. EdiKit is available as an EPROM ready to plug in, or as a ROM image on disc for loading into sideways RAM. Full details and order information are elsewhere in this issue.

CASSETTES VERSUS DISCS

Recent correspondence with a BEEBUG member has surprised us as it suggests that there are still a good many users of BBC micros who rely solely on cassettes for their storage requirements. Now this was not unreasonable when the Beeb was first launched with even the cheapest disc drives costing upwards of £400. Today, it is possible to come across brand new drives for as little as £50, and the enormous increase in ease of use that results is worth every penny.

Discs are also much more reliable, offering facilities such as immediate and random access to files which cannot be matched by any cassette system. Anyone contemplating the upgrade should note that they will require to ensure that their machine is fitted with a disc interface, in order to connect the disc drive unit to the micro. Details of interfaces and drives are given in our Retail Catalogue, and we have a Fact Sheet: *BBC Micro Disc Upgrade* which gives more detailed information on what is required. The small ads in BEEBUG can also be a useful source of add-ons.

512 CO-PROCESSOR

Checking through this month's batch of members' small ads I was struck by the number of *wanted* ads from readers seeking 512 co-processors. Fortunately, there were also several ads offering complete systems with a 512, so hopefully both sales and wants will find their needs satisfying. It would appear that there is a steadily growing interest in the facility to add MS-DOS capability to a BBC micro, and it would be nice to think that Robin Burton's regular column on the 512 in BEEBUG has had something to do with this.

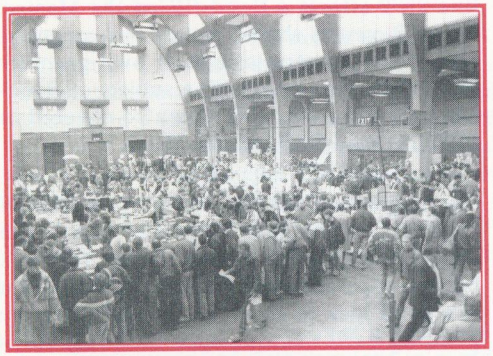
IF MUSIC BE THE FOOD OF LOVE

This issue marks the conclusion of Ian Waugh's short series of articles on programming in Ample, the language for Hybrid Technology's Music 5000. Like the 512 co-processor, this is another minority interest, but again one which has its devoted band of followers. If there is sufficient interest we are quite prepared to consider further articles by Ian, or even a regular column as with the 512. It's up to you.

News News News News News News

NEW SHOWS FOR ENTHUSIASTS

A new series of shows for computer enthusiasts has been launched in London. Known as the *All Formats Computer Fair* the first event was held on 10th February, and the second such event took place on 28th and 29th April. As the name implies, these fairs are intended to cover a range of different home computers. Starting with the next fair, there will be an Archimedes 'village' within the fair catering for Acorn enthusiasts. This show will take place on 9th and 10th June, at the New Horticultural Hall, Westminster, London from 10am till 5pm on the first day, and until 4pm on the second. Admission is priced at £3 per person. For further information contact John Riding on (0225) 447453 or Mike Hayes on (0457) 875229.



The first All Formats Computer Fair

COMPUTER SHOPPER SHOW EXPANDS

The *Computer Shopper Show* scheduled for December 1990 has been expanded to four full days and will now run from Thursday 6th December to Sunday 9th December. The venue has also been moved to the Wembley Conference Centre, such was the success of the inaugural show last December at Alexandra Palace. We understand that Acorn has booked a stand at the show, and BEEBUG will be there as well (stand J25). For more information contact Mike Cowley on 061-480 9811.

BBC ACORN USER SHOW

Just a reminder that this year's *BBC Acorn User Show*, usually considered the premier event for Acorn enthusiasts, is due to take place at the New Horticultural Hall, Westminster, from 7th to 9th September. Again, both Acorn and BEEBUG will have a presence (BEEBUG is on stand 74).

MINERVA MOVES HOUSE

Software house Minerva has announced a move to new offices. The new address is (unsurprisingly) Minerva Software, Minerva House, Baring Crescent, Exeter EX1 1TL. The telephone number remains the same at (0392) 437756, as does the fax number, (0392) 421762.

INDEFATIGABLE SUPERIOR

Yet another *Play it again Sam* collection of games has been released by software games house Superior. The latest offering is number 13 in the sequence and comprises *Barbarian II*, *Hyperball* (a brand new release), *Pandemonium* (a Peter Scott special now available for the first time on the Master or Electron), and a favourite classic, *Percy Penguin*, whose eponymous hero continues his battle against the Snobees. *Play it again Sam 13* costs £9.95 on dual cassette, £11.95 on 5.25" disc, and £14.95 on 3.5" disc for the Compact. Superior are at P.O.Box 6, Brigg, South Humberside DN20 9NH, tel. (0652) 58585.

AMPLINEX FOR MUSIC LOVERS

Readers of Ian Waugh's recent series of articles on *Programming in Ample*, the language of Hybrid Technology's Music 5000, may be interested to know of the existence of Amplinex, a group for devotees of this system. Amplinex operates as an information exchange, and publishes a regular bi-monthly disc with a combination of text, graphics and sound all fully integrated for the Music 5000 environment.

To join Amplinex, send a blank formatted DFS disc (40 track users send one double sided or two single sided), a self-addressed label and return postage, and £5.00 to Amplinex, 26 Arbor Lane, Winnersh, Berks RG11 5JD. Anyone contributing to Amplinex gets the next issue free.

AN OVATION FOR BEEBUG

Readers may be interested to know that BEEBUG's own full-featured DTP package for the Archimedes (called *Ovation*) is due for release about now. Priced at just £99.00 inc. VAT for BEEBUG and RISC USER members, the software should be available this June. For more information contact BEEBUG on (0727) 40303.

SplineText

David James offers a method of producing outline fonts on the BBC.

INTRODUCTION

SplineText is a utility program, designed to run on any BBC Model B, B+ or Master, together with a standard Epson-compatible printer. When run, the program asks the user for a string and then outputs it on the printer in Helvetica typestyle at any size up to 8"x7" per letter. The definitions are not stored as bit-maps like the standard 8x8 pixel BBC character set, but as a set of co-ordinates of points on the outline of each letter, which are then joined up either by straight lines or by a curve. I have used Roger Burg's spline routines from BEEBUG Vol.7 No.2 with a small step value for accuracy.

If you have a flood-fill routine in ROM (such as that in the Computer Concepts Graphics ROM or the Acorn GXR) or on disc, you can use this to fill the character. Remember, though, that large, filled characters take their toll on your printer ribbon! My solution to this is to remove my printer ribbon and use carbon paper, which gives excellent results. Each character is drawn on the screen and then sent to the printer by a special machine-code dump routine which directly accesses the screen for speed.

INSTRUCTIONS

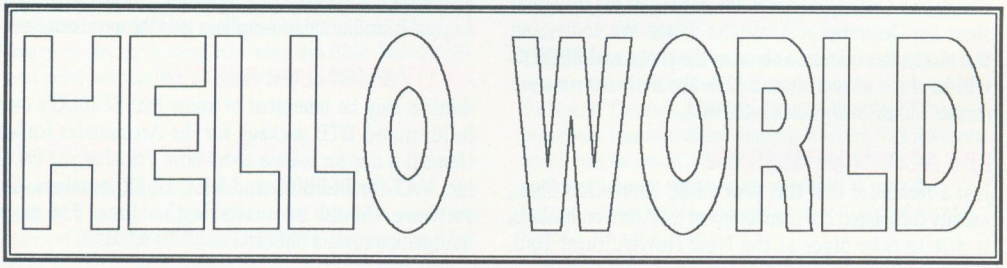
First of all, type in listing 1 and save it as "Spline1". This program assembles the code for the fast screen dump and then chains in the main program. The dump routine is 256 bytes long and, to speed up printing, uses a 768-byte buffer. The code is stored at &2F00, the buffer is between &2C00-&2EFF, and HIMEM is lowered to &2C00 to protect this area from the Basic stack. This configuration allows the program to run on all models. Model B owners should delete line 750, while for a Master you need not include lines 760-770.

Now enter the second listing and save it as "Spline2", taking care to enter the encoded character definitions carefully. *Do not* renumber the program or the computed RESTORE in line 1350 will not be correct. The PLOT command in line 2590 effects a flood-fill on the Master; model B owners should replace this command with one appropriate to their own graphics ROM if they have one. The subject of flood-filling for the model B was also covered in BEEBUG Vol.5 No.3 and Vol.6 No.3.

The program can now be run by typing CHAIN "Spline1". If you have made any mistakes in typing the data for any character, this will become apparent when you try to print it, so it is a good idea to check all the characters by printing each one out once the program is complete.

TECHNICAL INFORMATION

The shape of each character is defined by giving a sufficient number of points on the outline, along with the type of curve (spline or line)



needed to join them up. For example, the letter S is given as 2 S-shaped splines and 2 short straight lines to join them. The given points are restricted to a 55x55 grid, so we can store each point as 2 ASCII characters, representing the x and y positions on the grid. The values of these characters are found by the formula:

```
character=CHR$(48+2*coord)
```

but the interpolation between them isn't restricted to this grid.

The program uses the symmetry of letters to cut down the amount of data required in some cases; for example, we only need define the top-left hand corner of the letter 'O' to describe it fully. We also give the computer the width of the character, and the co-ordinates of one or more points to use as the start point for flood-filling (the exclamation mark is the only character for which we need 2 points).

CHARACTER DATA

The data statements given in the listing define capital letters A-Z, full stop, comma and exclamation mark. The first string of a data line is always 3 characters long, and supplies the x-axis of symmetry (zero means the letter is not symmetrical in the x-axis), the y-axis of symmetry (zero means not y-symmetrical), and the width of the character in comparative units. For example, 'A' would signify a width of 65 units, 'a' would be 97. These are not absolute values, but merely describe the relative widths of the characters to each other. If you are adding a new character to the list it is up to you to choose a relative width based on its relationship to characters already present.

The next string gives as many points as are necessary to flood-fill the letter properly (normally just one). Each string of points which then follows is preceded by 'S' (meaning use a spline to join them) or 'L' (use straight lines). The definition is terminated with 'E'.

The letter width as specified by the user when the program is run, refers to a letter of average width; so a seven-letter word with a letter width of one inch is not necessarily going to be seven inches long. Also, to aid spacing between letters, the width of each letter is rounded up to the nearest ninth of an inch - this has a negligible effect on how the printed phrase looks, however.

Listing 1

```
 10 REM Program Spline1
 20 REM Version B1.0
 30 REM Author  David James
 40 REM BEEBUG  June 1990
 50 REM Program Subject to copyright
 60 :
100 adr=&70:len=&76:dots=&77
110 S%=&2C00:C%=&2F00
120 FOR I%=0 TO 2 STEP 2
130 P%=C%:[
140 OPT I%
150 .setadr LDA adr:STA asl+1
160 LDA adr+1:STA asl+2:RTS
170 :
180 .asl ASL &FFFF,X:RTS
190 :
200 .block16 LDA #0:LDX #0
210 .L1 JSR asl:ROL A
220 CPX #&06:BNE L2
```

```
230 PHA:LDA #&78:CLC
240 ADC asl+1:STA asl+1
250 LDA #&02:ADC asl+2:STA asl+2
260 PLA
270 .L2 INX:INX
280 CPX #&10:BNE L1
290 .endblock16 LDY &78
300 STA (&74),Y:INY
310 STY &78:BNE rts
320 INC &75:.rts RTS
330 :
340 .block48 LDA #0:TAX:TAY
350 .L3 JSR asl:BCC L4
360 CLC:ADC (&72),Y
370 .L4 INX:INY
380 CPY dots:BNE L3
390 JMP endblock16
400 :
410 .data
420 EQUD &030C30C0
430 EQUD &10204080
440 EQUD &01020408
450 :
460 .blockdo LDA dots:CMP #&10
470 BNE not16
480 LDA #&07:.loop16 PHA
490 JSR setadr:JSR block16
500 PLA:SEC:SBC #1
510 BPL loop16:RTS
520 .not16
530 LDA#data MOD256:STA &72
540 LDA#data DIV256:STA &73
550 LDA dots:CMP #8
560 BNE not8
570 LDA &72:CLC
580 ADC #&04:STA &72
590 .not8 LDA #7
600 .loop48 PHA:JSR setadr
610 JSR block48
620 PLA:SEC:SBC #1:BPL loop48
630 RTS
640 :
650 .main
660 LDA #&00:STA &74:STA &78
670 LDA #S% DIV 256:STA &75
680 .mainloop JSR blockdo
690 LDA #&08:CLC:ADC adr:STA adr
700 LDA #&00:ADC adr+1:STA adr+1
710 DEC len:BNE mainloop:RTS
720 ]
730 NEXT
740 REM Model B owners delete line 750
750 CHAIN"Spline2"
```

```
760 *KEY0"LO.""Spline2""|M*T.|MFOR L%=
0 TO &1700 STEP 4:L%!&E00=L%!PAGE:NEXT:P
AGE=&E00|MO.|MRUN|M"
770 *FX 138 0 128
```

Listing 2

```
  10 REM Program Spline2
  20 REM Version B1.1
  30 REM Author  David James
  40 REM BEEBUG  June 1990
  50 REM Program Subject to copyright
  60 :
 100 code%=C%:store%=S%:main=code%+&9C:
adr=&70:len=&76:dots=&77
 110 DIM X%(25),Y%(25),apeX%(25),apeY%(
25)
 120 MODE 7:PROCheader:PROCgetparams
 130 MODE 0:HIMEM=&2C00
 140 VDU 2:*FX3,64
 150 FOR loop%=1 TO LEN(M$):CLS
 160 let$=MID$(M$,loop%,1):PROCdrawlett
er(let$)
 170 IF let$<>" " PROCdumpletter
 180 NEXT:VDU 3,7:END
 190 :
1000 DEF PROCheader
1010 FOR I%=0 TO 1
1020 VDU 157,129,141:PRINT"  SPLINETEXT
 V1.1 by David James"
1030 NEXT:VDU 31,0,24,157,129:PRINTSPC1
0"(C) BEEBUG 1990";:VDU 28,0,23,39,2
1040 ENDPROC
1050 :
1060 DEF PROCgetparams
1070 REPEAT PRINT'" Your message ";
1080 INPUT LINE M$:UNTIL LEN(M$)>0
1090 REPEAT PRINT'" Width of letters (i
nches) :";
1100 INPUT "" xsize:UNTIL xsize>0
1110 REPEAT PRINT'" Height of letters (
inches) :";
1120 INPUT "" ysize:UNTIL ysize>0
1130 REPEAT PRINT'" Letters filled (Y/N
) :";
1140 INPUT "" YN$:UNTIL INSTR("YyNn",YN
$)
1150 fill%=(YN$="Y" OR YN$="y"):xdpp=4
1160 IF xsize>7.1111 xsize=7.1111
1170 IF xsize<=(32/9) THEN xdpp=2
1180 IF xsize<=(16/9) THEN xdpp=1
1190 ydpp=3:IF ysize>8 ysize=8
1200 IF ysize<=(16/3) THEN ydpp=2
```

```
 1210 IF ysize<=(8/3) THEN ydpp=1
 1220 ysc=(ysize/ydpp)*1279*3/(8*28)
 1230 ENDPROC
 1240 :
 1250 DEF PROCscalex
 1260 xs=xsize:realx=xs*xmax/29
 1270 adjx=INT(0.5+realx*9)/9
 1280 xs=(adjx*xs)/realx
 1290 xsc=(1024*xs*9)/(16*xdpp*29)
 1300 xsc=(xmax*xsc-4)/xmax
 1310 ENDPROC
 1320 :
 1330 DEF PROCdrawletter(letter$)
 1340 IF letter$=" " THEN PROCls(108*xsi
ze):ENDPROC
 1350 RESTORE (2610+10*(INSTR("ABCDEFGHI
JKLMNOPQRSTUVWXYZ.!,",letter$)))
 1360 READ S$,fill$
 1370 XS=FNt(1):YS=FNt(2):xmax=FNt(3)
 1380 PROCscalex
 1390 REPEAT READ N$
 1400 IF N$<>"E" READ S$:L%=(LENS$)/2
 1410 IF N$="S" PROCspline ELSE IF N$="L
" PROCline
 1420 UNTIL N$="E"
 1430 IF fill% PROCfill
 1440 ENDPROC
 1450 :
 1460 DEF PROCline
 1470 PROCline1(0,1,0,1)
 1480 IF XS<>0 PROCline1(2*XS,-1,0,1)
 1490 IF YS<>0 PROCline1(0,1,2*YS,-1)
 1500 IF XS<>0 AND YS<>0 PROCline1(2*XS,
-1,2*YS,-1)
 1510 ENDPROC
 1520 :
 1530 DEF PROCline1(xc,xm,yc,ym)
 1540 x=FNd(1,xc,xm):y=FNd(2,yc,ym)
 1550 MOVE FNy(y),FNx(x)
 1560 FOR N%=1 TO L%-1
 1570 x=FNd(1+2*N%,xc,xm)
 1580 y=FNd(2*(N%+1),yc,ym)
 1590 DRAW FNy(y),FNx(x)
 1600 NEXT N%
 1610 ENDPROC
 1620 :
 1630 DEF PROCspline
 1640 PROCspline1(0,1,0,1)
 1650 IF XS<>0 PROCspline1(2*XS,-1,0,1)
 1660 IF YS<>0 PROCspline1(0,1,2*YS,-1)
 1670 IF XS<>0 AND YS<>0 PROCspline1(2*X
S,-1,2*YS,-1)
 1680 ENDPROC
 1690 :
 1700 DEF PROCspline1(xc,xm,yc,ym)
 1710 FOR N%=0 TO L%-1
 1720 x=FNd(1+2*N%,xc,xm):Y%(N%+1)=FNx(x
)
 1730 y=FNd(2*(N%+1),yc,ym):X%(N%+1)=FNy
(y)
 1740 NEXT N%
 1750 FOR J%=1 TO L%
 1760 apeX%(J%)=((X%(J%)*4)-X%(J%+1)-X%(
J%-1))/2
 1770 apeY%(J%)=((Y%(J%)*4)-Y%(J%+1)-Y%(
J%-1))/2
 1780 NEXT
 1790 MOVE X%(1),Y%(1)
 1800 FOR J%=1 TO L%-2
 1810 FOR n=0 TO 0.5 STEP .02
 1820 N=1-n:m=n*2:M=1-m
 1830 IF J%=1 PROCsimplebow(J%)
 1840 IF J%>1 PROChalfbow
 1850 NEXT:NEXT
 1860 FOR n=0.5 TO 1 STEP .02
 1870 PROCsimplebow(J%-1)
 1880 NEXT n
 1890 DRAW FNrx(X%(J%+1)),FNry(Y%(J%+1))
 1900 ENDPROC
 1910 :
 1920 DEF PROCsimplebow(J%)
 1930 DRAW FNrx(FNbowX(J%,n)),FNry(FNbow
Y(J%,n))
 1940 ENDPROC
 1950 :
 1960 DEF PROChalfbow
 1970 LOCAL BowX,BowX2,BowY,BowY2
 1980 BowX=m*FNbowX(J%,n)
 1990 BowX2=M*FNbowX(J%-1,n+.5)
 2000 BowY=m*FNbowY(J%,n)
 2010 BowY2=M*FNbowY(J%-1,n+.5)
 2020 DRAW FNrx(BowX+BowX2),FNry(BowY+Bo
wY2)
 2030 ENDPROC
 2040 :
 2050 DEF FNbowX(J%,n)
 2060 LOCAL X%,X2%,apeX%
 2070 X%=X%(J%):X2%=X%(J%+2)
 2080 apeX1%=apeX%(J%+1)
 2090 =((X%+(n*(apeX1%-X%)))*(1-n))+((ap
eX1%+(n*(X2%-apeX1%)))*n)
 2100 :
 2110 DEF FNbowY(J%,n)
 2120 LOCAL Y%,Y2%,apeY%
 2130 Y%=Y%(J%):Y2%=Y%(J%+2)
 2140 apeY1%=apeY%(J%+1)
```

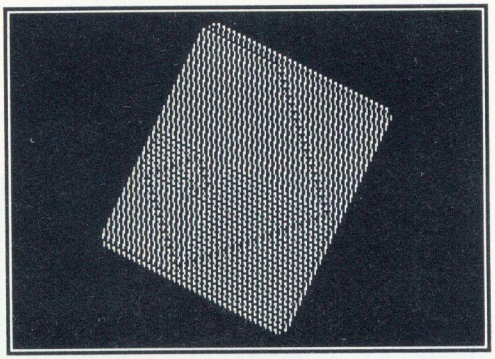
```
 2150 =((Y%+(n*(apeY1%-Y%)))*(1-n))+((ap
eY1%+(n*(Y2%-apeY1%)))*n)
 2160 :
 2170 DEF FNx(pos)=1023-xsc*pos
 2180 DEF FNy(pos)=ysc*pos
 2190 DEF FNrx(x):IF x<0 THEN =0
 2200 IF x>1279 THEN =1279 ELSE =x
 2210 DEF FNry(y):IF y<0 THEN =0
 2220 IF y>1023 THEN =1023 ELSE =y
 2230 :
 2240 DEF FNt(P%)=(ASC(MID$(S$,P%,1))-48
)/2
 2250 DEF FNd(P%,c,m)=c+m*FNt(P%)
 2260 :
 2270 DEF PROCdumpletter
 2280 ?dots=16/xdpp
 2290 lines%=INT(.99999999+(4+xmax*xsc)/
64)
 2300 len%=INT(.99999999+28*ysc/16)
 2310 PROCdump:PROCls(20*xsize)
 2320 ENDPROC
 2330 :
 2340 DEF PROCdump
 2350 FOR Y%=0 TO (xdpp*lines%)-1
 2360 IF xdpp=1 x%=&500*Y% ELSE IF xdpp=
2 x%=&280*Y% ELSE x%=&280*(Y% DIV 2)+4*(
Y% MOD 2)
 2370 !adr=&3000+x%:?len=len%
 2380 CALL main:PROCprint:PROCls(1)
 2390 IF xdpp=1 !adr=&3001+x%:?len=len%:
CALL main
 2400 PROCprint:PROCls(23)
 2410 NEXT Y%:ENDPROC
 2420 :
 2430 DEF PROCls(L%)
 2440 VDU 1,27,1,51,1,L%,1,13,1,10
 2450 ENDPROC
 2460 :
 2470 DEF PROCprint
 2480 VDU 1,27,1,90,1,(8*len%*ydpp) MOD&
100,1,(8*len%*ydpp) DIV&100
 2490 FOR I%=0 TO (8*len%-1)
 2500 FOR J%=1 TO ydpp
 2510 VDU 1,I%?store%
 2520 NEXT ,:ENDPROC
 2530 :
 2540 DEF PROCfill
 2550 S$=fill$:FOR F%=1 TO LENS$ STEP 2
 2560 fx%=FNx(FNt(F%))
 2570 fy%=FNy(FNt(F%+1))
 2580 REM Replace next line with call to
 your usual routine.
 2590 PLOT 133,fy%,fx%
 2600 NEXT:ENDPROC
 2610 :
 2620 DATA L0h,22,L,L;D;@101DgLg,L,LUFCL
C,E
 2630 DATA 00_,22,L,N1010gNg,S,NgVdZ£\[[
TWO,S,WO]H_@Z5N1,L,JR@R@ZJZ,S,JZMYOVMSJR
,L,L=@=@GLG,S,LGPEQBP?L=,E
 2640 DATA 0Ld,J2,S,dFb>X4K0>44>0L,S,UFR
AK>DAAF@L,L,dFUF,E
 2650 DATA 0Lb,22,L,0L01M1,S,M1X5£?bL,L,
@L@>F>,S,F>L@QERL,E
 2660 DATA 0L^,22,L,0M01^1^?@?@GZGZM,E
 2670 DATA 00Z,22,L,01@1@CVCVQ@Q@YZYZg0g
01,E
 2680 DATA 00d,J2,S,Y7R2K0>44>0L4Z>dKhXd
bZdR,S,UCQ?K>DAAF@LARDWKZRWUR,L,UCNCNNdN
d1Z1Y7,L,dRUR,E
 2690 DATA LLh,22,L,0N01@1@CLC,E
 2700 DATA 8L@,22,L,0M0181,E
 2710 DATA 00X,D2,L,HCHgXgXE,S,0E2;75D1Q
5V;XE,L,0E0I@I@C,S,@CA@D?G@HC,E
 2720 DATA 00d,22,L,010g@g@VOgbgPRd1R1DE
@A@101,E
 2730 DATA 00X,22,L,010g@g@AXAX101,E
 2740 DATA M0j,22,L,M1G1>P>1010gFgMJ,E
 2750 DATA 00£,22,L,010gAgOIOg£g£1O1ANA1
01,E
 2760 DATA KLf,K2,S,K0>44>0L,S,K?DBAG@L,
E
 2770 DATA 00_,22,L,LB@B@1010gLg,S,LgVd\
^_U\KVELB,L,IN@N@ZIZ,S,IZMXOTMPIN,E
 2780 DATA 00f,J2,S,K0>44>0L4Z>dKhXdbZfL
b>,L,b>f>f0K0,S,K?DBAG@LAQDVKYRVUQVLUGRB
K?,E
 2790 DATA 00£,22,L,FC@C@1010gOg,S,Og[a£
W^NXI,S,XI[F^A_;_3a1,L,a1P1,S,P1N=KBFC,L
,IM@M@YIY,S,IYMWOSMOIM,E
 2800 DATA 00\,F2,S,\WZ_SeFh9e2_0W0U2O9H
FDIAG=D<A=?A,S,0A2993F0S3Z9\BZISPFTCWE[H
\K[MW,L,\WMW,L,?A0A,E
 2810 DATA G0^,2f,L,G1?1?Y0Y0gGg,E
 2820 DATA I0b,2f,L,0D0g@g@D,S,@DB?I=,S,
0D1<65=1I0,E
 2830 DATA H0£,2f,L,HD@g0g?1H1,E
 2840 DATA T0x,2f,L,TQN1=10g?gFCMgTg,E
 2850 DATA KLf,22,L,KVDg0g=L,E
 2860 DATA K0f,2f,L,KSBg0gBEB1K1,E
 2870 DATA 00g,22,L,\g0g0YHY0?01\1\?D?\Y
\g,E
 2880 DATA 00>,22,L,01>1>?0?01,E
 2890 DATA 00>,225H,L,01>1>?0?01,L,4E0Y0
f>f>Y:E4E,E
 2900 DATA 00>,44,L,030?>?>3;0407303,E
```

Turmites

Grimble Gromble brings us another fascinating excursion into the world of recreational mathematics.

First there was LIFE, then there were Mathematical Worms (see BEEBUG Vol.7 No.4), and now we meet Turmites.

Some time ago I wrote a program, simulating Turing machines, which was really too complicated for publication. Since then an article has appeared in Scientific American (Computer Recreations by A.K.Dewdney p.124-127 September 1989) exploring the world of two-dimensional Turing machines and its close relative, the turmite.



A two-state turmite

Based on this I have produced a program encompassing the essential principles, which is simple to use, and produces a intriguing graphical display without any need to delve into abstruse mathematics.

The turmite lives alone in an infinite two-dimensional world of square cells. Its behaviour depends on two factors; the colour of the cell it is occupying and the state the turmite is in. Given these two values the turmite carries out a simple procedure: firstly it changes (or not) the colour of the occupied cell, next it makes a turn (or not) and advances by one cell, and finally it switches to a different state (or stays the same). This process is repeated ad infinitum.

The turmites simulated by the program listed here can use up to 8 colours (the limits of the BBC micro) and 26 states, though in theory these would be unlimited. Even so this will provide enormous scope for exploring these fascinating creatures. Of course, it is not necessary to use all of them. The main restriction is really the limited screen area which here kills the turmite when it reaches the screen boundaries. It is easiest to display (and manipulate) the relevant information as a table (corresponding to a Turing machine).

USING THE PROGRAM

The program should be typed in and saved before running. It first displays the decision table ready for editing. Each of the table entries defined by current cell colour (0-7 along the top) and present state (A-Z down the left side) contains three symbols. The first is the colour (0-7) to which the cell is to be changed. Next is the direction the turmite turns (forward ^, right >, back v and left <) before advancing one cell, and finally the next state (A-Z) the turmite adopts.

The following keys may be used in edit mode and a reminder is displayed:

0-7:	Set colour to leave cell.
Cursor keys:	Set direction to turn.
A-Z:	Set next state to adopt and move cursor to next column.
f0-f3:	Move cursor left, right, down and up respectively.
Return:	Move cursor to beginning of next row.
f4:	Set turmite running. World initially colour 0 and turmite facing to right of screen in state A.
f5:	Save decision table to named file.
f6:	Load decision table from named file.
f7:	Reset decision table to default. There are several reasons for using the default as you will see.
*:	Execute * command. Press Return alone to get back to edit mode.

Once the turmite is running (f4), pressing the 'S' will temporarily suspend the turmite's activity while the screen display is dumped to a file

called SCRn, where n starts at 0 and increases by one each time the 'S' key is pressed. The value of n starts at 0 for each run so if you want to keep results from more than one run, use the * facility to rename the files between runs.

```
Colour  0   1   2   3   4   5   6   7
State
    A  1>B 2<A 1^B 3^A 4^A 5^A 6^A 7^A
    B  3<A 4>B 2^B 3^B 4^B 5^B 6^B 7^B
    C  0^C 1^C 2^C 3^C 4^C 5^C 6^C 7^C
    D  0^D 1^D 2^D 3^D 4^D 5^D 6^D 7^D
    E  0^E 1^E 2^E 3^E 4^E 5^E 6^E 7^E
    F  0^F 1^F 2^F 3^F 4^F 5^F 6^F 7^F
    G  0^G 1^G 2^G 3^G 4^G 5^G 6^G 7^G
    H  0^H 1^H 2^H 3^H 4^H 5^H 6^H 7^H
    I  0^I 1^I 2^I 3^I 4^I 5^I 6^I 7^I
    J  0^J 1^J 2^J 3^J 4^J 5^J 6^J 7^J
    K  0^K 1^K 2^K 3^K 4^K 5^K 6^K 7^K
    L  0^L 1^L 2^L 3^L 4^L 5^L 6^L 7^L
    M  0^M 1^M 2^M 3^M 4^M 5^M 6^M 7^M
    N  0^N 1^N 2^N 3^N 4^N 5^N 6^N 7^N
    O  0^O 1^O 2^O 3^O 4^O 5^O 6^O 7^O
    P  0^P 1^P 2^P 3^P 4^P 5^P 6^P 7^P
    Q  0^Q 1^Q 2^Q 3^Q 4^Q 5^Q 6^Q 7^Q
    R  0^R 1^R 2^R 3^R 4^R 5^R 6^R 7^R
    S  0^S 1^S 2^S 3^S 4^S 5^S 6^S 7^S
    T  0^T 1^T 2^T 3^T 4^T 5^T 6^T 7^T
    U  0^U 1^U 2^U 3^U 4^U 5^U 6^U 7^U
    V  0^V 1^V 2^V 3^V 4^V 5^V 6^V 7^V
    W  0^W 1^W 2^W 3^W 4^W 5^W 6^W 7^W
    X  0^X 1^X 2^X 3^X 4^X 5^X 6^X 7^X
    Y  0^Y 1^Y 2^Y 3^Y 4^Y 5^Y 6^Y 7^Y
    Z  0^Z 1^Z 2^Z 3^Z 4^Z 5^Z 6^Z 7^Z
Colour:0-7 Direction:Arrows State:A-Z
Cursor Movement:f0-f3 and Return
Run:f4 Save:f5 Load:f6 Reset:f7 OS:*
```

The decision table

Pressing 'X' at any time will exit from run mode and return you to edit mode (the turmite display will then be lost).

If the turmite reaches the edge of the screen, a beep will sound and the cursor will reappear at the top left of the screen. Pressing any key will then return you to edit mode or 'S' will provide a final screen dump first.

Make sure you have room on the disc(s) for any dumps first. You can run some of these turmites for hours (if not days) and it's soul rending to request a dump only to have the display wiped before you've finished and be told 'Not enough room'. Each dump uses 20k (&50 sectors).

CREATING TURMITES

Well, that's the program, what about some turmites to play with? First, a convention. Since only some of the colours and states are normally used, any decision tables shown will only include the top left corner relevant to the turmite. For example:

```
1<A 0^B
1>A 1>A
```

is a two-state turmite (first row - state A, second row - state B) using two colours (first column - colour 0, second column - colour 1). As an example of the ease of entry, the above turmite is entered with the following sequence of key presses (no spaces):

```
1 < A 0 ^ B Return 1 > A 1 > A
```

where the <, ^ and > symbols represent the corresponding cursor keys at the top right of the keyboard. Now press f4 to see the turmite draw an unusual spiral.

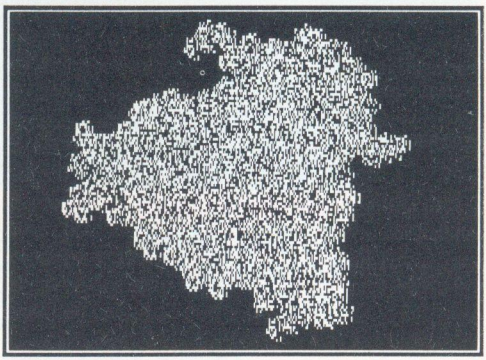
Even single-state turmites can display complex behaviour as demonstrated by:

```
1<A 0>A
```

Those of you who used mathematical worms will get a strong feeling of deja-vu. You should notice several resemblances in behaviour despite the radically different environment and rules employed. Perhaps even more curious is another single-state turmite:

```
1>A 2>A 3<A 0<A
```

which produces a symmetrical design (it runs for hour after hour before hitting the screen edge so be warned; use 'X' if you can't take any more).



A complex three-state turmite

There are only two more pieces of advice I'm going to give. Firstly, unlike worms, turmites can cover the same ground repeatedly. This means they can get stuck in loops, some of which can be quite long in execution. If you're not sure if this has happened, move on to a different one. There are so many possibilities, why waste time on a doubtful one?

Lastly, you may want to see just where a turmite has been and this can be difficult since sometimes it colours cells black and the background is black

too. You can overcome this by using one extra colour and doubling the number of states. Take the first two-state turmite mentioned above:

```
1<A 0^B
1>A 1>A
```

Increase the colour numbers by one and shift the decisions one column to the right. For each used state, in column 0 enter 1v and the character of an unused state (different for each used state). Now for each new state, go across the columns entering the colour corresponding to the column, 'v', and the used state for which this state was employed. You should end up with:

```
1vC 2<A 1^B
1vD 2>A 2>A
0vA 1vA 2vA
0vB 1vB 2vB
```

It sounds complicated but the reasoning is quite straightforward as you will see if you manually trace the action of this and the original through a few steps, bearing in mind that in the new display the colour numbers are one greater than in the old display.

```
   10 REM Program Turmite
   20 REM Version B1.1
   30 REM Author  Grimble Gromble
   40 REM BEEBUG  June 1990
   50 REM Program subject to copyright
   60 :
  100 PROCinit:PROCreset
  110 ON ERROR MODE 7:PROCerror:IF end%
END
  120 REPEAT
  130 MODE 1:Q%=FNe
  140 IF Q%=1 MODE2:PROCturmite
  150 IF Q%=2 PROCsave
  160 IF Q%=3 PROCload
  170 IF Q%=4 PROCreset
  180 IF Q%=5 MODE 7:PROCos
  190 UNTIL FALSE
  200 :
 1000 DEF PROCinit
 1010 LOCAL I%,T%:N%=0
 1020 DIM c%(7,25),d%(7,25),s%(7,25)
 1030 DIM x%(3),y%(3),z%(3),d$(3)
 1040 FOR I%=0 TO 3
 1050 READ x%(I%),y%(I%),z%(I%),d$(I%)
 1060 NEXT
 1070 q%=-67:k%=-82:f%=128:*FX225,128
 1080 ENDPROC
 1090 :
 1100 DATA 8,0,3,"^"
 1110 DATA 0,-8,1,">"
 1120 DATA -8,0,2,"v"
 1130 DATA 0,8,0,"<"
 1140 :
 1150 DEF PROCreset
 1160 LOCAL C%,S%
 1170 FOR C%=0 TO 7:FOR S%=0 TO 25
 1180 c%(C%,S%)=C%:d%(C%,S%)=0:s%(C%,S%)
=S%
 1190 NEXT S%,C%
 1200 ENDPROC
 1210 :
 1220 DEF PROCerror
 1230 LOCAL V%:end%=FALSE
 1240 CLOSE#0:*FX4,0
 1250 IF ERR<>17 REPORT:PRINT" at line "
;ERL:end%=TRUE:ENDPROC
 1260 PRINT"Continue (Y/N/*)?";
 1270 REPEAT
 1280 V%=INSTR("NnYy*",GET$)
 1290 UNTIL V%
 1300 IF V%=5 THEN PROCos
 1310 IF V%>2 THEN ENDPROC
 1320 PRINT "N"
 1330 end%=TRUE:*FX225,1
 1340 ENDPROC
 1350 :
 1360 DEF PROCos
 1370 LOCAL G$:CLS:VDU14
 1380 REPEAT
 1390 INPUT LINE'"*"G$:OSCLI(G$)
 1400 UNTIL G$=""
 1410 ENDPROC
 1420 :
 1430 DEF FNe
 1440 LOCAL C%,S%,G%,Q%:Q%=0
 1450 PROCshow:*FX4,2
 1460 *FX 21,0
 1470 REPEAT
 1480 PRINTTAB(4*C%+8,S%+2);
 1490 G%=GET
 1500 IF G%>=48 AND G%<=55 THEN PROCcol(
G%-48)
 1510 IF G%>=f%+12 AND G%<=f%+15 THEN PR
OCdir(z%(G%-(f%+12)))
 1520 IF G%>=65 AND G%<=90 THEN PROCstat
e(G%-65)
 1530 IF G%=13 THEN C%=0:S%=(S%+1)MOD26
 1540 IF G%=ASC("*") THEN Q%=5
 1550 IF G%=f%    C%=(C%+7)MOD8
 1560 IF G%=f%+1 C%=(C%+1)MOD8
 1570 IF G%=f%+2 S%=(S%+1)MOD26
 1580 IF G%=f%+3 S%=(S%+25)MOD26
 1590 IF G%>=f%+4 AND G%<=f%+7 Q%=G%-(f%
+3)
```

```
 1600 UNTIL Q%:*FX4,0
 1610 =Q%
 1620 :
 1630 DEF PROCshow
 1640 LOCAL C%,S%:CLS
 1650 PRINT"Colour"'"State"
 1660 FOR C%=0 TO 7
 1670 PRINTTAB(4*C%+9,0);C%;
 1680 NEXT
 1690 FOR S%=0 TO 25
 1700 PRINTTAB(5,S%+2)CHR$(ASC("A")+S%);
 1710 FOR C%=0 TO 7
 1720 PRINTTAB(4*C%+8,S%+2);c%(C%,S%);d$
(d%(C%,S%));CHR$(ASC("A")+s%(C%,S%));
 1730 NEXT C%,S%
 1740 PRINTTAB(0,29)"Colour:0-7 Directio
n:Arrows State:A-Z";
 1750 PRINTTAB(0,30)"Cursor Movement:f0-
f3 and Return";
 1760 PRINTTAB(0,31)"Run:f4 Save:f5 Load
:f6 Reset:f7 OS:*";
 1770 ENDPROC
 1780 :
 1790 DEF PROCcol(V%)
 1800 c%(C%,S%)=V%:PRINT;V%;
 1810 ENDPROC
 1820 :
 1830 DEF PROCdir(V%)
 1840 d%(C%,S%)=V%
 1850 PRINTTAB(4*C%+9,S%+2)d$(V%);
 1860 ENDPROC
 1870 :
 1880 DEF PROCstate(V%)
 1890 s%(C%,S%)=V%
 1900 PRINTTAB(4*C%+10,S%+2)CHR$(ASC("A"
)+V%);
 1910 C%=(C%+1)MOD8
 1920 IF C%=0 THEN S%=(S%+1)MOD26
 1930 ENDPROC
 1940 :
 1950 DEF PROCturmite
 1960 LOCAL C%,D%,S%,X%,Y%,exit%
 1970 X%=640:Y%=512:exit%=FALSE
 1980 VDU 23,1,0;0;0;0;
 1990 REPEAT
 2000 C%=POINT(X%,Y%)
 2010 IF C%=-1 OR INKEY(q%) THEN exit%=T
RUE:GOTO2080
 2020 IF INKEY(k%) THEN PROCscr
 2030 GCOL 0,c%(C%,S%)
 2040 PLOT 69,X%,Y%:PLOT 65,0,4
 2050 D%=(D%+d%(C%,S%))MOD4
 2060 X%=X%+x%(D%):Y%=Y%+y%(D%)
 2070 S%=s%(C%,S%)
 2080 UNTIL exit%
 2090 VDU 23,1,1;0;0;0;
 2100 IF C%=-1 THEN VDU 7 ELSE ENDPROC
 2110 C%=GET AND &DF
 2120 IF C%=83 PROCscr
 2130 ENDPROC
 2140 :
 2150 DEF PROCscr
 2160 OSCLI("SAVE SCR"+STR$(N%)+" FFFF30
00 +5000")
 2170 N%=N%+1
 2180 ENDPROC
 2190 :
 2200 DEF PROCsave
 2210 LOCAL F$,F%,C%,S%
 2220 F$=FNfile("Save")
 2230 IF F$="" THEN ENDPROC
 2240 F%=OPENIN(F$):REM Check file exist
s
 2250 IF F% THEN CLOSE#F%:IF FNcheck THE
N ENDPROC
 2260 F%=OPENOUT(F$)
 2270 FOR C%=0 TO 7:FOR S%=0 TO 25
 2280 PRINT#F%,c%(C%,S%),d%(C%,S%),s%(C%
,S%)
 2290 NEXT S%,C%
 2300 CLOSE#F%
 2310 ENDPROC
 2320 :
 2330 DEF PROCload
 2340 LOCAL F$,F%,C%,S%
 2350 F$=FNfile("Load")
 2360 IF F$="" THEN ENDPROC
 2370 F%=OPENIN(F$)
 2380 FOR C%=0 TO 7
 2390 FOR S%=0 TO 25
 2400 INPUT#F%,c%(C%,S%),d%(C%,S%),s%(C%
,S%)
 2410 NEXT
 2420 NEXT
 2430 CLOSE#F%
 2440 ENDPROC
 2450 :
 2460 DEF FNfile(P$)
 2470 LOCAL F$:CLS
 2480 PRINT P$;" File: ";
 2490 INPUT""F$
 2500 =F$
 2510 :
 2520 DEF FNcheck
 2530 LOCAL G$
 2540 PRINT'"File exists. Overwrite (Y/N
)?";
 2550 REPEAT
 2560 G%=GET AND &DF:G$=CHR$(G%)
 2570 UNTIL INSTR("YN",G$)
 2580 PRINT G$
 2590 =(G$="N")
```

Storing Variables and Procedures in SWR on a Model B

Expand your programs by using sideways RAM for storing additional procedures, functions and variables. Norman Smith explains.

GENERAL

In BEEBUG Vol.6 No.6, Graham Crow demonstrated how to store and overlay Basic procedures, functions and variables on the Master 128 and Compact using sideways RAM. These programs, of course, could not be used directly on the model B as its lacks the SRDATA, SRWRITE, and SRREAD commands, and the multiple sideways RAM banks, but the principle was interesting and useful enough to investigate the possibility of producing similar features for the earlier machine when fitted with sideways RAM.

The accompanying programs are all based on the original ones and produce the same results using the Aries B-12 ROM/RAM board with one bank of 16k RAM which is fitted in slot 0 on this board. With a change in value of one of the variables, there would not appear to be any reason why other ROM/RAM boards with the RAM in slot 15 or any other position should not work correctly. Of course, the extra memory is restricted to 16k but this can still be a worthwhile extension to the normal user RAM area.

ENTERING THE PROGRAMS

The programs, which have had the amendments and additions commented, should be typed in and saved, before running. For convenience, the complete programs (not just the amendments) are listed here. Listing 1 is the procedure/function overlay program, while listing 2 comprises the variable storage program. Each program not only implements the relevant routines but also provides a complete working example.

PROCEDURE/FUNCTION OVERLAYS

The principle on which this is based is that a number of functions or procedures are saved individually as files on disc. Normally these files will have the same names as the functions or procedures which they contain. When an overlay program is run, the additional procedures and functions are loaded into sideways RAM (their names are included in a DATA statement in the main program). When any one of these functions or procedures is called it is dynamically loaded from sideways RAM into user RAM and executed. Each function or procedure called in this way overlays (overwrites) any previously copied into user RAM.

To experiment with this technique, type in and save, with the names shown, the following procedure definitions (taken from the original article):

```
10000 :
10010 REM "CHART"
10020 :
10030 DEF PROCchart
10040 PRINT'"CHART OVERLAY"
10050 ENDPROC
```

Save this file as "Chart"

```
10000 :
10010 REM "EDITOR"
10020 :
10030 DEF PROCeditor
10040 PRINT'"EDITOR OVERLAY"
10050 ENDPROC
```

Save this file as "Editor"

```
10000 :
10010 REM "FILER"
10020 :
10030 DEF PROCfiler
10040 PRINT'"FILER OVERLAY"
10050 ENDPROC
```

Save this as "Filer"

Now run the program from listing 1. This loads the three procedures above into sideways RAM. From the menu which is then displayed on the screen, selecting any one of the three options causes the corresponding procedure to be loaded from sideways RAM to main memory before it is executed.

To use the technique with your own programs follow the format of listing 1. Note that LOMEM (in line 110) must be set so that it will always be above the program plus the longest overlay. Line 1050 is set to the number of overlays, and line 1110 is a DATA statement containing the names of the overlay files. PROCinit sets up the sideways RAM ready for use, and PROCloadSWR loads the overlay routines into sideways RAM. PROCappend loads an overlay from sideways RAM adding it to the end of the main program. PROCexecute calls the relevant procedure depending upon the menu choice (PROCmenu).

VARIABLES IN SIDEWAYS RAM

The program in listing 2 shows how the same principle may be applied to the storage of variables in sideways RAM, though this is generally of less use. It stores an example of each of five different data types (byte, double byte, integer, real, and string) in sideways RAM, and then reads the same data back again. This is handled by five short procedures (starting at line 1320) and five functions (at line 1610 onwards). The demonstration is performed by PROCdemo at line 1140. When storing a value in sideways RAM it is up to the user to determine where this should go, just as with the use of indirection operators in Basic. This should become clear if you examine the coding of the demonstration routine.

PROGRAM DETAILS

With due acknowledgement to Graham Crow, these routines follow the original format and line numbers of his programs. The necessary changes are to the mode - mode 7 has been used in the example, setting the slot number using the variable B% to the required bank, and replacing the OSCLI calls to SRWRITE and SRREAD by procedures calling a short machine code program assembled at &A00. Using only one bank of sideways RAM makes the SRDATA call superfluous anyway.

The procedures PROCsrwrite and PROCsrread poke the original parameters into a work area starting at &70 to be used by the machine code which is then activated by *CALL mcopy*. These procedure calls replace the original OSCLI calls.

PROCEDURE OVERLAY PROGRAM

There are few changes apart from that to the mode, and the addition of B% at line 1020 (set to the required sideways RAM slot number), the procedure calls and definitions, and PROCassemble. Also, at line 1380 in this program, &8000 must be added to the variable ad%.

VARIABLE STORAGE PROGRAM

The same changes as the procedure overlay program have been made plus the following points. The variables A%,X%,Y%,O% or P% should not be used (line 1060 et seq) for variable A, - the original A% has been replaced by V% in this program. The addresses in line 1220 - PROCs("Any string",&4000) - and line 1290 - PRINT...FNs(&4000) - should have the &4000 reduced in value as this address is at the top limit of the sideways RAM - &3800 has been used in the program. Of course, any or all of these data addresses may be changed to suit personal requirements.

CONCLUSION

The overlay system is very effective compared with disc-based overlay systems. When applied to a BBC model B it does not appear to slow down its response to any great extent when compared with procedures or functions held in normal user RAM. It should be noted that to speed up the response even further (subject to memory availability, the value of LOMEM and the length of each procedure or function), several compatible routines with consecutive line numbers could be saved under one filename and then loaded as one file. The first named one should be the master procedure, which then calls the others as appropriate.

A possible alternative for RAM/ROM extension boards with several banks of SWR would be to add an extra parameter to PROCsrwrite and PROCsrread containing the pertinent values of B% to address each active SWR slot.

Listing 1

```
10 REM Program SWR Procedure Overlays
20 REM Version B0.3
30 REM Author Graham Crow
40 REM Amended for BBC Model B
50 REM and Aries B-20 RAM/ROM board
60 REM Author N.L.Smith
```

```
  70 REM BEEBUG June 1990
  80 REM Program subject to copyright
  90 :
 100 MODE 7
 110 LOMEM=&4000
 120 REM LOMEM must be > top of main
 130 REM program + longest overlay
 140 PROCinit
 145 PROCassemble
 150 PROCloadSWR
 160 REPEAT
 170 PROCmenu
 180 PROCappend
 190 PROCexecute
 200 UNTIL FALSE
 210 :
1000 DEF PROCinit
1010 REM set up SWR bank in slot 0
1020 B%=0 :REM adjust to suit SWR board
1030 T%=TOP :REM Procs load at T%-2
1040 S%=0 :REM SWR address for overlays
1050 N%=3:REM No of overlays
1060 DIM F$(N%):REM overlay filenames
1070 DIM L%(N%):REM overlay lengths
1080 DIM R%(N%):REM SWR address
1090 RESTORE 1110
1100 FOR J%=1 TO N%:READ F$(J%):NEXT
1110 DATA CHART,FILER,EDITOR
1120 ENDPROC
1130 :
1140 DEF PROCloadSWR
1150 REM loads overlays into SWR
1160 FOR J%=1 TO N%:OSCLI("LOAD "+F$(J%
)+" "+STR$~T%)
1170 Z%=OPENUP F$(J%)
1180 L%(J%)=EXT#Z%:CLOSE#Z%
1190 PRINT'"LOADING "F$(J%)" (";L%(J%);
" bytes) INTO SWR AT"
1195 PRINT"ADDRESS ";~(32768+S%)
1200 PROCsrwrite
1210 R%(J%)=S%:S%=S%+L%(J%)
1220 NEXT
1230 PRINT';N%;" overlays now in SWR. P
ress any key for demo...":IF GET
1240 ENDPROC
1250 :
1260 DEF PROCmenu
1270 CLS:PRINT'"MAIN MENU"
1280 PRINT'"1...CHART"'"2...FILER"
1290 PRINT"3...EDITOR"'"4...END"''"Whic
h? ";
1300 REPEAT G%=GET-48
1310 UNTIL G%>0 AND G%<5
1320 IF G%=4 CLS:?(T%-2)=&D:?(T%-1)=&FF
:END
1330 ENDPROC
1340 :
1350 DEF PROCappend
1360 REM appends overlay at TOP-2
1370 len%=L%(G%):ad%=R%(G%)
1380 ad%=ad%+&8000:PROCsrread:ENDPROC
1390 :
1400 DEF PROCexecute:CLS
1410 IF G%=1 PROCchart
1420 IF G%=2 PROCfiler
1430 IF G%=3 PROCeditor
1440 PRINT"Press any key ":IF GET
1450 ENDPROC
1460 :
1470 DEF PROCsrwrite
1480 ?&70=TOP MOD 256:?&71=TOP DIV 256:
?&72=S% MOD 256:?&73=&80+(S% DIV 256):?&
74=B%:?&75=L%(J%) DIV 256:?&76=L%(J%) MO
D 256:IF ?&75=0 ?&77=0 ELSE ?&77=1
1490 IF ?&76=0 ?&79=0 ELSE ?&79=1
1500 CALL mcopy
1510 ENDPROC
1520 :
1530 DEF PROCsrread
1540 ?&70=ad% MOD 256:?&71=ad% DIV 256:
?&72=(TOP-2) MOD 256:?&73=(TOP-2) DIV 25
6:?&74=B%:?&75=len% DIV 256:?&76=len% MO
D 256:IF ?&75=0 ?&77=0 ELSE ?&77=1
1550 IF ?&76=0 ?&79=0 ELSE ?&79=1
1560 CALL mcopy
1570 ENDPROC
1580 :
1590 DEF PROCassemble
1600 FOR opt%=0TO2STEP2:P%=&A00
1610 [OPT opt%
1620 .mcopy LDA &F4:STA &78:LDA &74
1630 STA &F4:STA &FE30:LDX #0
1640 CPX &77:BEQ smallp
1650 .pagelp LDY #0
1660 .bytelp1 LDA (&70),Y:STA (&72),Y
1670 INY:BNE bytelp1
1680 INX:INC &71:INC &73
1690 CPX &75:BNE pagelp
1700 .smallp LDY #0
1710 CPY &79:BEQ fin
1720 .bytelp2  LDA (&70),Y:STA (&72),Y
1730 INY:CPY &76:BNE bytelp2
1740 .fin LDA &78:STA &F4:STA &FE30:RTS
1750 ]NEXT:ENDPROC
```

Listing 2

```
  10 REM Program SWR Variable storage
  20 REM Version B0.7
  30 REM Author Graham Crow
  40 REM Amended for BBC Model B
  50 REM and Aries B-20 RAM/ROM board
  60 REM Author N.L.Smith
  70 REM BEEBUG June 1990
  80 REM Program subject to copyright
```

```
  90 :
 100 MODE 7
 110 A=0:REM assign A at start
 120 PROCinit
 125 PROCassemble
 130 PROCdemo
 140 END
 150 :
1000 DEF PROCinit
1010 REM set up SWR bank
1020 REM SWR in slot 0
1030 B%=0:REM adjust to suit board
1040 :
1050 REM assign address of variable A
1060 REM to some variable eg V%
1070 V%=LOMEM+3
1080 :
1090 REM reserve string storage space
1100 REM must be longer than string+1
1110 max%=100:DIM M% max%
1120 ENDPROC
1130 :
1140 DEF PROCdemo
1150 PRINT'"DEMONSTRATION OF WRITE/READ
SIDEWAYS RAM"'
1160 :
1170 REM write data to SWR
1180 PROCb(&E,&0)       :REM single byte
1190 PROCbb(&FE99,&1000)   :REM two byte
1200 PROCi(1234567,&2000)     :REM integ
1210 PROCr(-1234.567,&3000)   :REM real
1220 PROCs("Any string",&3800):REM str
1230 :
1240 REM read data from SWR
1250 PRINT"Byte:"TAB(13)"&";~FNb(0)
1260 PRINT"Double-byte: &";~FNbb(&1000)
1270 PRINT"Integer:"TAB(13);FNi(&2000)
1280 PRINT"Real:"TAB(13);FNr(&3000)
1290 PRINT"String:"TAB(13)FNs(&3800)
1300 ENDPROC
1310 :
1320 DEF PROCb(num%,ad%):REM write byte
1330 ?V%=num%
1340 PROCsrwrite(V%,1)
1350 ENDPROC
1360 :
1370 DEF PROCbb(num%,ad%):REM write 2by
te
1380 LOCAL b1%,b2%
1390 b1%=num%DIV256:b2%=num%MOD256
1400 PROCb(b1%,ad%):PROCb(b2%,ad%+1)
1410 ENDPROC
1420 :
1430 DEF PROCi(num%,ad%):REM write int
1440 !V%=num%
1450 PROCsrwrite(V%,4)
1460 ENDPROC
1470 :
1480 DEF PROCr(num,ad%):REM write real
1490 A=num
1500 PROCsrwrite(V%,5)
1510 ENDPROC
1520 :
1530 DEF PROCs(str$,ad%):REM write str
1540 LOCAL len%
1550 $M%=str$
1560 len%=LEN(str$)+1:PROCb(len%,ad%)
1570 ad%=ad%+1
1580 PROCsrwrite(M%,len%)
1590 ENDPROC
1600 :
1610 DEF FNb(ad%):REM read byte
1620 PROCsrread(V%,1)
1630 =?V%
1640 :
1650 DEF FNbb(ad%):REM read 2 byte
1660 LOCAL b1%,b2%
1670 b1%=FNb(ad%)*256:b2%=FNb(ad%+1)
1680 =b1%+b2%
1690 :
1700 DEF FNi(ad%):REM read integer
1710 PROCsrread(V%,4)
1720 =!V%
1730 :
1740 DEF FNr(ad%):REM read real
1750 PROCsrread(V%,5)
1760 =A
1770 :
1780 DEF FNs(ad%):REM read string
1790 LOCAL len%
1800 len%=FNb(ad%):ad%=ad%+1
1810 PROCsrread(M%,len%)
1820 =$M%
1830 :
1840 DEF PROCsrwrite(Q%,len%)
1850 ?&70=Q% MOD 256:?&71=Q% DIV 256:?&
72=ad% MOD 256:?&73=&80+(ad% DIV 256):?&
74=B%:?&75=len%:CALL mcopy
1860 ENDPROC
1870 :
1880 DEF PROCsrread(Q%,len%)
1890 ?&70=ad% MOD 256:?&71=&80+(ad% DIV
 256):?&72=Q% MOD 256:?&73=Q% DIV 256:?&
74=B%:?&75=len%:CALL mcopy
1900 ENDPROC
1910 :
1920 DEF PROCassemble
1930 FOR opt%=0TO2STEP2:P%=&A00
1940 [OPT opt%
1950 .mcopy LDA &F4:STA &76:LDA &74
1960 STA &F4:STA &FE30:LDX #0
1970 LDY #0
1980 .lp LDA (&70),Y:STA (&72),Y
1990 INY:CPY &75:BNE lp
2000 .fin LDA &76:STA &F4:STA &FE30:RTS
2010 ]NEXT:ENDPROC
```

Fog Index

David Jupe shows how to tell the difference between a legal document and an article in the tabloid press!

These days we have word processors and spelling checkers, but there is a general lack of other software which can be used to examine and analyse text. The program described below attempts to overcome that by analysing the complexity of a document.

A word processor is often used to prepare a speech or an article to be presented to other people. How easily they will understand it will depend on how complicated it is. Difficulty in understanding will depend on the nature of the subject and the logical complexity. It will also depend on the length and complexity of the words and sentences used. If the sentences are long, with many subordinate clauses, they will be more difficult to follow than if they are short. Similarly if big words are used these are also more difficult to take in than short words.

In America they have come up with the idea of a "Fog Index" which can be calculated for a text. The more difficult the text is to understand, the higher the index. The program FOG listed here will determine the Fog Index of a piece of text held as an ASCII file on disc. It is written to ignore Wordwise control characters, but may be used with text produced by any other word processor so long as it is an ASCII file. This can usually be achieved by spooling the file, or from View by saving the file without rulers (either by deleting them first or by using WRITE with the markers set to exclude them).

The program uses a combination of word count, sentence count, and big-word count to determine the Fog Index. Big words are, with a few exceptions, those of three syllables or more. Where a word ends in "ed" or "es" this is not included in the syllable count. It is not easy in a short program to determine how many syllables there are in a word; the program uses the rough rule of equating the number of vowel groups (the letter y being treated as a vowel) with the number of syllables, and this seems to be quite adequate. Occasionally it does not work, and words such as "sometimes" and "therefore" are treated as big words though they really are not. However, this may be balanced out by words such as "area" and "denying" which have three syllables but are counted as small words. Words ending in "e" and "le" are treated in a special way.

The formula for the Fog Index is:

$$0.4 \times ((T/S) + (B \times 100)/T)$$

where:

T = total word count
S = sentence count
B = big-word count

Splitting a long sentence into two (or more) shorter ones will reduce the Fog Index. Replacing big words by shorter ones will also have the same effect. Of course, one has to be careful not to destroy the meaning of the text. Examples of word replacement might be "nice" for "delightful", "chew" for "masticate" and so on. On many occasions a big word cannot be avoided.

Text with a Fog Index value below 10 is very easy to understand; with 12 it is moderately difficult; with 16 and above it is sure to put people to sleep. Legal documents with words like "hereinafter", "aforesaid" and "nevertheless" tend to have a high Fog Index. The tabloid press tends to have a low Fog Index.

Type the program in and save it as FOG. To find the Fog Index of a text file load the program, then place the disc containing the text file into the current drive. Typing RUN will cause the catalogue to be displayed, and the name of the file may now be entered. As the text is analysed it will be displayed on the screen, a sentence at a time, with the word count for the sentence on the following line. Big words are highlighted in red. At the end of the analysis the word, sentence, and big-word counts are shown, together with the Fog Index.

The end of a sentence is determined by the presence of a full stop in a string of characters

containing a letter. Question marks, exclamation marks and semicolons have the same effect and are treated as sentence terminators. Abbreviations which contain or are terminated by a full stop can fool the program into believing that it has found a short sentence, so it is best to edit these out first.

Sentences are only counted if they contain at least one word. Words are strings of letters, terminated most frequently by a space or a punctuation mark. A word may include an apostrophe or hyphen; thus "o'clock" or "semi-detached" would be counted as one word. In numeric or monetary strings the full stop is treated as a decimal point and not as a sentence terminator. Expressions such as "ABCD1234", "+5.23", "£1,000" and "-" are not counted as words.

Examination of a long text may be interrupted by using the Escape key, and an analysis will be carried out on the complete sentences read so far. An analysis really needs at least 100 words to give a sensible result and, although the calculations will still be carried out on shorter texts, the results for a word count of less than 100 should not be treated as reliable.

Because of the lack of regularity in the English language, a program such as this is always capable of tuning and improvement. However there must, in the end, be a compromise between speed, accuracy and size.

This article, ignoring the title and formula, has a Fog Index of 11.5 so it shouldn't be too difficult to follow!

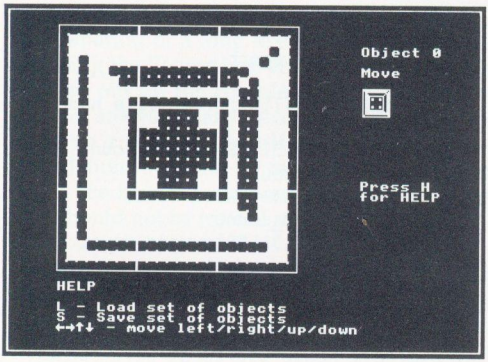
```
  10 REM Program Fog
  20 REM Version B1.0
  30 REM Author  David R. Jupe
  40 REM BEEBUG  June 1990
  50 REM Program subject to Copyright
  60 :
 100 sentence_count=0
 110 ON ERROR GOTO 1930
 120 *KEY0RUN|M
 130 MODE7
 140 *CAT
 150 PROCinitialize
 160 PRINT
 170 REPEAT
 180 PROCreadchar
 190 UNTIL EOF#(channel)
 200 PROCword
 210 CLOSE#(channel)
 220 PROCreport
 230 END
 240 :
1000 DEF PROCreadchar
1010 REM Read next character from text
1020 value=BGET#(channel)
1030 char$=CHR$(value)
1040 printable=(value>&20 AND value<&7F
)
1050 sentence=((INSTR(".",char$) AND (l
etter_status OR word$="")) OR (INSTR(";!
?",char$)))
1060 word=(value=&20 OR value=&A OR val
ue=&D)
1070 punctuation=INSTR("()'[]{}"",:-.;!
?",char$)>0
1080 control=(value<&21 AND NOT (value=
2 OR word))
1090 letter=((value>&40 AND value<&5B)
OR (value>&60 AND value<&7B)):IF letter
THEN letter_status=TRUE
1100 IF printable AND NOT (letter OR pu
nctuation) THEN word_status=FALSE
1110 IF printable THEN PROCaddchar
1120 IF sentence THEN PROCword: PROCsen
tence ELSE IF value=2 THEN PROCword: PRO
Cskip ELSE IF word OR control OR value>&
7E THEN PROCword
1130 ENDPROC
1140 :
1150 DEF PROCaddchar
1160 REM Add the character to the word
1170 word$=word$+char$
1180 char_status=INSTR("aeiouyAEIOUY",c
har$)>0
1190 IF char_status AND vowel_status=FA
LSE THEN syllable_count=syllable_count+1
1200 IF char_status THEN vowel_status=T
RUE ELSE vowel_status=FALSE
1210 ENDPROC
1220 :
1230 DEF PROCskip
1240 REM Skip Wordwise controls
1250 REPEAT
1260 value=BGET#(channel)
1270 UNTIL EOF#(channel) OR value=7 OR
value=&D
1280 ENDPROC
1290 :
1300 DEF PROCword
1310 REM Analyse a word
1320 IF word$="" THEN PROCreset:ENDPROC
1330 print_word$=word$
1340 char$=RIGHT$(word$,1)
```

```
 1350 IF INSTR("()'[]{}"",:-.!?;",char$)
 AND LEN(word$)>1 THEN word$=LEFT$(word$
,LEN(word$)-1)
 1360 IF NOT letter_status THEN word_sta
tus=FALSE
 1370 IF LEN(word$)>1: suffix$=RIGHT$(wo
rd$,2): left$=LEFT$(suffix$,1): right$=R
IGHT$(suffix$,1): IF (left$="e" OR left$
="E") AND (right$="d" OR right$="D" OR r
ight$="s" OR right$="S") THEN syllable_c
ount=syllable_count-1
 1380 IF LEN(word$)>1 AND (RIGHT$(word$,
1)="e" OR RIGHT$(word$,1)="E") THEN end$
=RIGHT$(word$,2): left$=LEFT$(end$,1): I
F INSTR("aeiouyAEIOUY",left$)=0 THEN syl
lable_count=syllable_count-1
 1390 len=LEN(word$): IF len>2 THEN righ
t3$=RIGHT$(word$,3): IF right3$="ied" OR
  right3$="IED" OR  right3$="ies" OR  ri
ght3$="IES" THEN syllable_count=syllable
_count+1
 1400 IF len>3 THEN right4$=RIGHT$(word$
,4): right1$=RIGHT$(right4$,1): IF right
1$="S" OR right1$="s" THEN right3$=LEFT$
(right4$,3)
 1410 IF len>2 THEN IF (RIGHT$(right3$,2
)="le" OR RIGHT$(right3$,2)="LE") AND NO
T INSTR("aeiouyAEIOUY",LEFT$(right3$,1))
 THEN syllable_count=syllable_count+1
 1420 IF LEN(word$)>1 AND syllable_count
<1 THEN syllable_count=1
 1430 IF word_status AND syllable_count>
2 THEN big_word_count=big_word_count+1
 1440 IF word_status AND LEN(word$)>0 TH
EN word_count=word_count+1
 1450 IF (line+LEN(print_word$))>39 THEN
 PRINT:line=0
 1460 IF LEN(print_word$)>0 THEN line=(l
ine+1+LEN(print_word$))MOD40
 1470 IF syllable_count>2 PRINTCHR$(129)
; ELSE IF LEN(print_word$)>0 PRINTCHR$(1
35);
 1480 PRINTprint_word$;
 1490 PROCreset
 1500 ENDPROC
 1510 :
 1520 DEF PROCsentence
 1530 REM End of sentence reached
 1540 PRINT'CHR$(131)"(",word_count," wo
rds)"'
 1550 IF word_count>0 THEN sentence_coun
t=sentence_count+1
 1560 total_word_count=total_word_count+
word_count
 1570 word_count=0
 1580 total_big_word_count=total_big_wor
d_count+big_word_count
 1590 big_word_count=0
 1600 line=0
 1610 ENDPROC
 1620 :
 1630 DEF PROCinitialize
 1640 @%=1
 1650 INPUT'"Enter text file name: "file
$
 1660 channel=OPENIN(file$)
 1670 sentence_count=0:total_word_count=
0:word_count=0:total_big_word_count=0:bi
g_word_count=0
 1680 line=0
 1690 PROCreset
 1700 ENDPROC
 1710 :
 1720 DEF PROCreset
 1730 REM Initialize variables
 1740 syllable_count=0:word$=""
 1750 vowel_status=FALSE:word_status=TRU
E:letter_status=FALSE
 1760 ENDPROC
 1770 :
 1780 DEF PROCreport
 1790 REM Final report
 1800 @%=6
 1810 PRINT''CHR$(131)"Text file name ="
CHR$(135)file$
 1820 PRINTCHR$(131)"Sentence count ="CH
R$(135)sentence_count
 1830 PRINTCHR$(131)"Word count      ="CH
R$(135)total_word_count
 1840 PRINTCHR$(131)"Big-word count ="CH
R$(135)total_big_word_count
 1850 @%=&20108
 1860 IF sentence_count>0 THEN index=0.4
*((total_word_count/sentence_count)+(tot
al_big_word_count*100/total_word_count))
 ELSE index=0
 1870 PRINTCHR$(131)"FOG index       =";:
IF index<10 THEN PRINTCHR$(130); ELSE IF
 index>12 THEN PRINTCHR$(129); ELSE PRIN
TCHR$(131);
 1880 PRINTindex"  ";
 1890 IF index<10 THEN PRINT"(Safe value
)"' ELSE IF index>12 THEN PRINT"(TOO HIG
H)"' ELSE PRINT"(Moderate)"'
 1900 IF total_word_count<100 THEN PRINT
CHR$(129)CHR$(136)"Word count too small"
'
 1910 ENDPROC
 1920 :
 1930 REM Errors
 1940 IF ERR=222 THEN PRINT"File not fou
nd"''"PRESS Key f0 to re-run"'
 1950 IF ERR=17 THEN PROCreport
 1960 IF ERR<>222 AND ERR<>17 THEN REPOR
T:PRINT" at line ";ERL
 1970 CLOSE#0:END
```

A Versatile Character Editor

Jonathan Ribbens describes a comprehensive and versatile character definer with many useful features.

One of the useful features of BBC Basic is the facility for the user to define his own characters using the VDU23 statement. This not only allows characters which are not part of the machine's default character set to be defined as required, but also allows graphics characters to be created which have many applications, particularly as icons in all manner of programs (they are used in the program listed here), and in games. Furthermore, groups of more than one character can be defined in order to construct larger graphics objects.



Using the character editor

However, any new character definition has to be drawn out on paper and manually converted into a set of eight numbers to be included in the VDU23 instruction. This can be tedious, and lacks flexibility if you need to re-edit a character once defined, and it does not help when trying to define groups of two or more characters.

That's where the comprehensive character editor listed here will come in useful. It allows a block of three by three characters (here called an *object*) to be defined visually on screen together, and up to ninety characters can be defined altogether at one time. Character definitions can be saved in a form which permits further editing if required, but also creates a spooled file containing the complete VDU23 definitions for direct incorporation in any program of your own.

To use the program, simply type it in and save it before running it. The screen display shows an enlarged block of three by three characters, with the normal-sized version alongside (initially blank of course), details of the current editing mode, and space at the foot of the screen of other information or messages.

The functions of the character editor are all single key operations as shown in table 1.

Cursor keys	Move cursor
D	Draw mode
E	Erase mode
M	Move mode
Return	Plot a single pixel
Delete	Delete a single pixel
<	Select previous object
>	Select next object
Escape	Quit/abort function
H	Help
W	Wipe object
R	Read character values from pre-defined character
I	Input VDU23 values from keyboard
C	Copy one object to another
S	Save objects
L	Load objects

Table 1. List of key functions

There are three basic modes of operation. In *Draw* mode, using the cursor keys will leave a trail of set pixels. In *Erase* mode, this will leave a trail of unset pixels, and in *Move* mode it leaves pixels as they are unless you press Return or Delete (see table). Any function can be aborted by pressing Escape.

Each *object* of three by three characters is numbered on the screen starting at zero. The keys marked '<' and '>' (note, use these keys without pressing Shift) will move backwards or

forwards to the next object. When the *Copy* option is used, the specified object (by number) is copied into the current object definition on screen. *Wipe* simply clears the current object (after due warning).

As well as defining characters on screen, definitions can also be input in two other ways, by entering the VDU23 values directly from the keyboard, or by specifying a character code which has already been defined in the computer with a VDU23 statement. Thus, if you run the character editor immediately after running a program which has defined its own characters, you can easily pick these up in the character editor.

If you select the *Save* option, you are asked whether you want to save the characters in a file which can be re-edited, or in a file to be EXECed. If you select a file to be re-edited, you can load the characters back in again later (by using the *Load* option).

If you select the EXEC file option, you are asked for the Basic line numbers to give each definition, the line number increment, the file name, and the start VDU23 ASCII value (this will normally be 224 - see the user guide). This is so that the designs can be readily incorporated into your own programs. You may also choose whether or not to include characters which are completely blank. This is so that you can have a four by four object, for example, and be able to have it on the grid as you would see it in your program. If you choose to include blank characters, then all 90 definitions will be produced, and you may subsequently need to delete any which are not wanted.

Finally, the program incorporates its own *Help* system. When invoked, just press the space bar to cycle through the help information.

USING THE CHARACTER DEFINITIONS

Character definitions which have been spooled out to a file (see above) may be incorporated into any other program using the *EXEC command. Once defined, a character may be displayed on the screen in character position (x,y) using:

```
PRINTTAB(x,y) CHR$(n)
```

If multiple-character objects are to be used, it is best to pre-construct the object. For example:

```
OB1$=CHR$(n1)+CHR$(n2)+CHR$(10)+CHR$(8)
+CHR$(8)+CHR$(n3)+CHR$(n4)
```

would create a two by two object from character definitions n1, n2, n3 and n4. The sequence 10, 8, 8 after the first two characters moves down one line and back two positions ready for the second row of two characters. These codes appear in the list of VDU codes in the user guide. For more precise positioning, VDU5 allows characters to be placed at the graphics cursor using the MOVE instruction.

PROGRAM NOTES

The procedures and functions used in the program are as follows:

PROCinit	Initialise screen, characters and arrays
PROCquit	Close files and reset Escape and cursor keys
PROCedit	Main loop
PROCdisplay(n)	Display object 'n' on the grid
PROCplot(n)	Set or unset pixel at current position
PROCmini(x,y,n)	Set or unset pixel on real-size grid
PROCsave	Select format for saving
PROCclear	Clear message area
PROCload	Load a character file
PROCsavea	Save in format for re-editing
PROCsaveb	Save in format for EXECing
PROCsaveb1	Called by PROCsaveb
PROChelp	Display help messages
PROCread	Read an already defined character to the grid
PROCtile(n,x,y)	Display a single 'tile' (after Reading or Input)
PROCinput	Input VDU23 values
PROCwipe	Wipe the current object
PROCcopy	Copy one object to another
FNchar(x,y)	Returns character number for piece of grid
FNsure	Returns TRUE or FALSE according to whether the user wants to quit or not
FNin	Input routine

```
   10 REM Program CharEdit
   20 REM Version B1.1
   30 REM Author  Jonathan Ribbens
   40 REM BEEBUG  June 1990
   50 REM Program subject to copyright
   60 :
  100 ON ERROR MODE7:PROCquit:IF ERR=17
END ELSE REPORT:PRINT" at line ";ERL:END
  110 MODE4
  120 PROCinit:PROCedit
  130 PROCquit:MODE7:END
  140 :
 1000 DEF PROCinit
 1010 VDU19,0,4;0;
 1020 VDU23,1,0;0;0;0;
 1030 VDU23,128,129,0;0;0;129
 1040 VDU23,129,-1;-1;-1;-1;
 1050 VDU23,130,-1,128,128,128,128,128,1
28,129
 1060 VDU23,131,-1,1,1,1,1,1,1,129
 1070 VDU23,132,129,1,1,1,1,1,1,-1
 1080 VDU23,133,129,128,128,128,128,128,
128,-1
 1090 VDU23,134,-1,0;0;0;129
 1100 VDU23,135,129,1,1,1,1,1,1,129
 1110 VDU23,136,129,0;0;0;-1
 1120 VDU23,137,129,128,128,128,128,128,
128,129
 1130 VDU23,138,0,32,96,254,254,96,32,0
 1140 VDU23,139,0,8,12,254,254,12,8,0
 1150 VDU23,140,48,120,252,48,48,48,48,0
 1160 VDU23,141,48,48,48,48,252,120,48,0
 1170 DIM ob%(9,2,23)
 1180 *FX4,1
 1190 *FX229,1
 1200 ENDPROC
 1210 :
 1220 DEF PROCquit
 1230 *FX4
 1240 *FX229
 1250 CLOSE#0
 1260 ENDPROC
 1270 :
 1280 DEF PROCedit:LOCAL K%
 1290 P%=0:X%=0:Y%=0:M%=0
 1300 PROCdisplay(P%)
 1310 VDU31,30,15:PRINT"Press H"
 1320 VDU31,30,16:PRINT"for HELP"
 1330 REPEAT:REPEAT:*FX15 1
 1340 COLOUR1:PRINTTAB(30,4);:IF M%=0:PR
INT"Move " ELSE IF M%=1 PRINT"Draw " ELS
E IF M%=2 PRINT"Erase"
 1350 VDU31,X%,Y%:VDU23,1,1;0;0;0;
 1360 K%=GET:IF K%>96 AND K%<123 K%=K% A
ND &DF
 1370 IFK%=136 IFX%>0 X%=X%-1
 1380 IFK%=137 IFX%<23 X%=X%+1
 1390 IFK%=138 IFY%<23 Y%=Y%+1
 1400 IFK%=139 IFY%>0 Y%=Y%-1
 1410 IFK%=46 IFP%<9 P%=P%+1:PROCdisplay
(P%)
 1420 IFK%=44 IFP%>0 P%=P%-1:PROCdisplay
(P%)
 1430 IFK%=68 M%=1
 1440 IFK%=77 M%=0
 1450 IFK%=69 M%=2
 1460 IFK%=13  OR M%=1 PROCplot(1)
 1470 IFK%=127 OR M%=2 PROCplot(0)
 1480 IFK%=83 PROCsave
 1490 IFK%=76 PROCload
 1500 IFK%=72 PROChelp
 1510 IFK%=82 PROCread
 1520 IFK%=73 PROCinput
 1530 IFK%=87 PROCwipe
 1540 IFK%=67 PROCcopy
 1550 UNTIL K%=27:UNTIL FNsure
 1560 ENDPROC
 1570 :
 1580 DEF PROCdisplay(P%)
 1590 LOCAL X%,Y%,Z%,V%,W%,U%
 1600 VDU23,1,0;0;0;0;
 1610 FOR X%=0TO2:FOR Y%=0TO2
 1620 FOR Z%=0TO7:V%=ob%(P%,X%,Y%*8+Z%)
 1630 VDU31,X%*8,Y%*8+Z%
 1640 FOR W%=0TO7:U%=(V% AND (2^W%))
 1650 COLOUR1:IF U% VDU129 ELSE VDUFNcha
r(W%,Z%)
 1660 PROCmini(X%*8+W%,Y%*8+Z%,SGNU%)
 1670 NEXTW%,Z%,Y%,X%:COLOUR1
 1680 PRINTTAB(30,2);"Object ";P%
 1690 VDU23,1,1;0;0;0;30:ENDPROC
 1700 :
 1710 DEF FNchar(X%,Y%)
 1720 IFX%=0 IFY%=0 =130
 1730 IFX%=7 IFY%=0 =131
 1740 IFX%=7 IFY%=7 =132
 1750 IFX%=0 IFY%=7 =133
 1760 IFY%=0 =134
 1770 IFX%=7 =135
 1780 IFY%=7 =136
 1790 IFX%=0 =137
 1800 =128
 1810 :
 1820 DEF PROCplot(V%):VDU31,X%,Y%
 1830 COLOUR1:IF V%=0 VDUFNchar(X%MOD8,Y
%MOD8) ELSE VDU129
 1840 PROCmini(X%,Y%,V%)
 1850 IF V%=0 ob%(P%,X% DIV8,Y%)=ob%(P%,
X% DIV8,Y%) AND (NOT(2^(X% MOD8))):ENDPR
OC
```

```
 1860 ob%(P%,X% DIV8,Y%)=ob%(P%,X% DIV8,
Y%) OR 2^(X% MOD8):ENDPROC
 1870 :
 1880 DEF PROCmini(X%,Y%,C%)
 1890 GCOL0,C%
 1900 PLOT69,964+X%*4,832-Y%*4
 1910 ENDPROC
 1920 :
 1930 DEF PROCsave:LOCAL A%
 1940 COLOUR1:VDU31,0,25
 1950 PRINT"SAVE"'''"Save in  A. format f
or re-editing"'SPC(9);"B. format for EXE
Cing"
 1960 *FX15 1
 1970 REPEAT A%=GET AND &DF:UNTIL A%=65
OR A%=66 OR A%=27
 1980 PROCclear
 1990 IFA%=65 PROCsavea
 2000 IFA%=66 PROCsaveb
 2010 PROCclear:ENDPROC
 2020 :
 2030 DEF PROCclear:LOCAL I%
 2040 FOR I%=24TO30
 2050 PRINTTAB(0,I%);SPC40;:NEXT
 2060 PRINTTAB(0,31);SPC39;
 2070 VDU30:ENDPROC
 2080 :
 2090 DEF FNsure:LOCAL A%
 2100 COLOUR1:VDU31,0,25
 2110 *FX15 1
 2120 PRINT"QUIT"'''"Press any key to qui
t, or Escape to     carry on editing"
 2130 A%=(GET<>27):PROCclear:=A%
 2140 :
 2150 DEF PROCload:LOCAL F$,F%
 2160 COLOUR1:VDU31,0,25
 2170 PRINT"LOAD"'''"Enter filename: ";
 2180 F$=FNin:IF F$="" PROCclear:ENDPROC
 2190 F%=OPENIN(F$)
 2200 IF F%=0 PRINT'"File not found - Pr
ess any key":F%=GET:PROCclear:ENDPROC
 2210 PRINT'"Loading ..."
 2220 FOR P%=0TO9:FOR X%=0TO2:FOR Y%=0TO
23
 2230 ob%(P%,X%,Y%)=BGET#F%:NEXT,,
 2240 CLOSE#F%:PROCclear:P%=0
 2250 X%=0:Y%=0:M%=0:PROCdisplay(P%)
 2260 ENDPROC
 2270 :
 2280 DEF FNin:LOCAL S$,A%:S$=""
 2290 VDU23,1,1;0;0;0;
 2300 REPEAT:*FX15 1
 2310 A%=GET
 2320 IF A%=127 IF LENS$>0 S$=LEFT$(S$,L
ENS$-1):VDU127
 2330 IF A%>31 AND A%<127 IF LENS$<30:S$
=S$+CHR$A%:VDUA%
 2340 UNTIL A%=13 OR A%=27:VDU23,1,0;0;0
;0;
 2350 IF A%=27 PRINTSTRING$(LENS$,CHR$8)
;STRING$(LENS$," ");STRING$(LENS$,CHR$8)
;"<Escape>":=""
 2360 PRINT:=S$
 2370 :
 2380 DEF PROCsavea
 2390 LOCAL P%,X%,Y%,F$,F%:VDU31,0,25
 2400 PRINT"SAVE IN FORMAT FOR RE-EDITIN
G"'"Enter filename: ";:F$=FNin:IF F$=""
:ENDPROC
 2410 IF F$="":ENDPROC
 2420 PRINT'"Saving ..."
 2430 F%=OPENOUT(F$)
 2440 FOR P%=0TO9:FOR X%=0TO2:FOR Y%=0TO
23
 2450 BPUT#F%,ob%(P%,X%,Y%):NEXT,,
 2460 CLOSE#F%:ENDPROC
 2470 :
 2480 DEF PROCsaveb:VDU31,0,25
 2490 LOCAL P%,X%,Y%,F$,F%,A%,B%,D%
 2500 PRINT"SAVE IN FORMAT FOR EXECing"'
'"Enter filename: ";:F$=FNin:IF F$="":EN
DPROC
 2510 INPUT"Enter first line number: "L%
 2520 INPUT"Enter increment: "I%
 2530 INPUT"Start character: "C%
 2540 PRINT"Omit blank Characters (Y/N):
";
 2550 REPEAT:A%=GET AND &DF:UNTIL A%=89
OR A%=78:A%=(A%=78)
 2560 PRINTTAB(0,VPOS)"Saving ...";SPC25
;:VDU21
 2570 OSCLI"SPOOL "+F$
 2580 FOR P%=0TO9:FOR Y%=0TO2
 2590 FOR X%=0TO2:D%=FALSE
 2600 FOR B%=0TO7
 2610 IF ob%(P%,X%,Y%*8+B%)>0 D%=TRUE
 2620 NEXTB%:IF D% OR A% PROCsaveb1:C%=C
%+1
 2630 NEXT X%,Y%,P%
 2640 *SPOOL
 2650 VDU6
 2660 ENDPROC
 2670 :
 2680 DEF PROCsaveb1:LOCAL A%
 2690 PRINT;L%;"VDU23,";C%;:L%=L%+I%
 2700 FOR A%=0TO7
 2710 V%=ob%(P%,X%,Y%*8+A%)
 2720 V%=(V% AND1)*128+(V% AND2)*32+(V%
```

```
AND4)*8+(V% AND8)*2+(V% AND16)/2+(V% AND
 32)/8+(V% AND64)/32+(V% AND128)/128
 2730 PRINT ;",";V%;
 2740 NEXT A%:PRINT
 2750 ENDPROC
 2760 :
 2770 DEF PROChelp:LOCAL A%
 2780 COLOUR1:VDU31,0,25
 2790 PRINT"HELP"'''"L - Load set of obje
cts"'"S - Save set of objects";CHR$138;
CHR$139;CHR$140;CHR$141;" - move left/ri
ght/up/down"
 2800 *FX15 1
 2810 A%=GET:PROCclear:VDU31,0,25
 2820 PRINT"HELP"'
 2830 PRINT"Return - Plot a point"
 2840 PRINT"Delete - Erase a point"
 2850 PRINT"M - Move mode"
 2860 *FX15 1
 2870 A%=GET:PROCclear:VDU31,0,25
 2880 PRINT"HELP"'
 2890 PRINT"E - Erase mode"
 2900 PRINT"D - Draw mode"
 2910 PRINT"I - Input VDU 23 values"
 2920 *FX15 1
 2930 A%=GET:PROCclear:VDU31,0,25
 2940 PRINT"HELP"'
 2950 PRINT"R - Read from character in m
emory"
 2960 PRINT"C - Copy from object"
 2970 PRINT"W - Wipe object"
 2980 *FX15 1
 2990 A%=GET:PROCclear:VDU31,0,25
 3000 PRINT"HELP"'
 3010 PRINT"< - Move to previous charact
er"
 3020 PRINT"> - Move to next character"
 3030 PRINT"I - Input VDU 23 values"
 3040 *FX15 1
 3050 A%=GET:PROCclear:ENDPROC
 3060 :
 3070 DEF PROCread:COLOUR1:VDU31,0,25
 3080 A%=X% DIV8:B%=Y% DIV8:PRINT"READ"'
 3090 LOCAL X%,Y%,C%,D%,V%
 3100 PRINT"Enter character to read: ";:
C%=VALFNin:IF C%=0 PROCclear:ENDPROC
 3110 ?&70=X%:?&71=Y%:?&72=A%
 3120 ?&900=C%:X%=0:Y%=9:A%=10:CALL&FFF1
 3130 X%=?&70:Y%=?&71:A%=?&72
 3140 FOR D%=0TO7
 3150 V%=?(&901+D%)
 3160 V%=(V% AND1)*128+(V% AND2)*32+(V%
AND4)*8+(V% AND8)*2+(V% AND16)/2+(V% AND
32)/8+(V% AND64)/32+(V% AND128)/128
 3170 ob%(P%,A%,B%*8+D%)=V%
 3180 NEXT D%:PROCclear
 3190 PROCtile(P%,A%,B%):ENDPROC
 3200 :
 3210 DEF PROCtile(P%,A%,B%)
 3220 LOCAL U%,V%,W%,Z%
 3230 FOR Z%=0TO7:V%=ob%(P%,A%,B%*8+Z%)
 3240 VDU31,A%*8,B%*8+Z%
 3250 FOR W%=0TO7:U%=(V% AND (2^W%))
 3260 COLOUR1:IF U%:VDU129 ELSE VDUFNcha
r(W%,Z%)
 3270 PROCmini(A%*8+W%,B%*8+Z%,SGNU%)
 3280 NEXT W%,Z%:ENDPROC
 3290 :
 3300 DEF PROCinput:COLOUR1:VDU31,0,25
 3310 LOCAL A%,A$,B%,D%,V%
 3320 A%=X% DIV8:B%=Y% DIV8
 3330 PRINT"INPUT"
 3340 FOR D%=0TO7
 3350 VDU31,0,27
 3360 PRINT"Enter value for row ";D%+1;"
: ";
 3370 VDU31,22,27,32,32,32,31,22,27
 3380 A$=FNin:IF A$="":PROCclear:D%=8:NE
XT D%:ENDPROC ELSE V%=VALA$
 3390 V%=(V% AND1)*128+(V% AND2)*32+(V%
AND4)*8+(V% AND8)*2+(V% AND16)/2+(V% AND
32)/8+(V% AND64)/32+(V% AND128)/128
 3400 ob%(P%,A%,B%*8+D%)=V%
 3410 NEXT D%:PROCclear
 3420 PROCtile(P%,A%,B%):ENDPROC
 3430 :
 3440 DEF PROCwipe:LOCAL A%,B%
 3450 COLOUR1:VDU31,0,25
 3460 PRINT"WIPE"'''"Press any key to wip
e, or Escape to     carry on editing"
 3470 *FX15 1
 3480 A%=GET:PROCclear:IF A%<>27:FOR A%=
0TO2:FOR B%=0TO23:ob%(P%,A%,B%)=0:NEXT B
%,A%:PROCdisplay(P%)
 3490 ENDPROC
 3500 :
 3510 DEF PROCcopy:LOCAL A%,B%,F%
 3520 COLOUR1:VDU31,0,25
 3530 INPUT"COPY"'''"Copy from which obje
ct: "F%
 3540 IF F%<0 OR F%>9 PROCclear:ENDPROC
 3550 PRINT'"Press any key to copy, or E
scape to     carry on editing"
 3560 *FX15 1
 3570 A%=GET:PROCclear:IF A%<>27:FOR A%=
0TO2:FOR B%=0TO23:ob%(P%,A%,B%)=ob%(F%,A
%,B%):NEXT B%,A%:PROCdisplay(P%)
 3580 ENDPROC
```

Practical Assembler (Part 2)

by Bernard Hill

As promised in last month's article this time we are looking at error handling. We shall also be having a look at macros, and how they can be implemented in BBC assembler.

ERROR HANDLING

There is a special instruction on the 6502 microprocessor to enable any operating system built around this microprocessor to handle errors in a simple way. This instruction is BRK, and has an opcode (the value it becomes when assembled) of 0 - so you can always substitute the shorter "BRK" instruction for any "EQUB 0". When the 6502 executes a BRK instruction it jumps to a routine whose address is always kept in &FFFE-&FFFF. Don't confuse BRK with the Break key, however. The Break key is hard-wired to the 'reset' line on the microprocessor, and when this is connected a jump is made to the location whose address is held in &FFFC-&FFFD. As a consequence, therefore, any 6502 system must have ROM rather than RAM at the top end of its memory so that these values can be 'burnt in' to the four locations &FFFC-&FFFF, and this also explains why CALL !&FFFC is equivalent to pressing Break.

But to get back to BRK. The routine in the BBC operating system which handles BRK has also been coded to produce a standard error number (in Basic called ERR) and error message from the bytes which follow the BRK instruction. These must be coded as follows:

```
BRK
one byte ERR number
error message
one zero byte to terminate the message
```

For example, if we have a piece of code which is to perform integer division, then we would probably want to generate an error if the divisor were zero. This might look like:

```
LDA divisor
BNE over  \ jump over error if <>0
BRK
EQUB 255
EQUS "Division by zero attempted"
EQUB 0
.over
....
```

This is a very simple way of programming error-handling, and examples can be seen in my articles *FIND and *DFIND (BEEBUG Vol.8 Nos.7 and 10) where bad commands and non-existent files produce error messages. The error number you use is your concern and usually doesn't matter unless you want to trap it in Basic with an ON ERROR statement. It's a good idea, though, to use error numbers outside the range of values used by Basic (1 to 45), the operating system and the DFS (189 to 254). Popular choices are 0 and 255.

One exception to this is Escape handling. When a machine code program runs it cannot be interrupted unless the author arranges it to be, and the easy way to do this is to test for Escape having being pressed. Provided Escape has not been disabled with *FX229,1 then, on Escape, the interrupt handler arranges that the top bit of location &FF will be set. This can therefore be very easily tested for in a program at convenient places with a piece of code like this:

```
BIT &FF \ test location &FF
BPL over \ top bit was not set
BRK
EQUB 17
EQUS "Escape"
EQUB 0
.over
....
```

You'll find this device also in the articles mentioned above, where it is also necessary before issuing the error to close down the open file.

ERRORS IN SIDEWAYS RAM

As I warned in last month's article, when producing a program for sideways RAM, error

trapping is not quite so easy. This is due to the fact that when processing the BRK, the operating system pages the language ROM back in so that we would be reading a spurious error message from the language ROM rather than the one where the error occurred. The usual solution is to copy the BRK, the error number and message into the bottom of the stack (&100) and JMP there to execute it:

```
\ Escape handling...
BIT &FF:BPL over
LDA #0:STA &100 \ the BRK instruction
LDX #0
.lp LDA error,X:STA &101,X
BEQ fin \ the message ends with 0 byte
INX:JMP lp
.fin JMP &100 \ perform the error
.error EQUB 17:EQUS "Escape"+CHR$0
```

MACROS

In most assembler systems available on microcomputers the concept of a 'macro' is well established. This is a shorthand method for producing in the code a commonly-executed set of assembler instructions. For instance, the saving of A, X and Y registers on the stack is accomplished with:

```
PHA:TXA:PHA:TYA:PHA
```

or, in 6512 (Basic IV) we can use:

```
PHA:PHX:PHY
```

This set could perhaps be called "PUSHALL".

In BBC Basic we have the ability to define assembler macros by jumping back into Basic and there are two ways of doing this, using OPT or EQUS:

1. The normal use of OPT is to set the assembler option number, and it is common to use the variable 'opt', 'pass', 'pas%' or similar. So if we have the line:

```
OPT FNpushall
```

in our code, and:

```
DEF FNpushall
[ OPT opt:PHA:TXA:PHA:TYA:PHA:]
=opt
```

(assuming our assembly loop variable is called 'opt'), then when assembled this will produce the five instructions defined in FNpushall. Note the 'dummy' assignment to return 'opt' as the function value.

2. Using EQUS is very similar, and there is little to choose between this and OPT, except that this method needs only one change if you ever change the name of your assembler loop variable:

```
[ ... assembler code ...
EQUS FNpushall
... ]

DEF FNpushall
[OPT opt:PHA:TXA:PHA:TYA:PHA:]
=""
```

Note that we assign the dummy function in this case to the null string to avoid the EQUS producing spurious assembled code.

It is important to realise that this idea does not save any space in the final assembly. We still produce 5 bytes of code whether we explicitly include the PHA...PHA in the middle of our program or use FNpushall. But it does save us space in the program which contains the source code, and as a refinement we can even include parameters to the function. For instance, it is common in some programs to store integers in two consecutive bytes and so there is a need to copy them to other locations. I frequently use the macro:

```
DEF FNcopy2(a,b)
[ OPT opt:PHA:LDA a:STA b
LDA a+1:STA b+1:PLA:]
=""
```

which copies two consecutive bytes to another two consecutive bytes. We could call it with, for example:

```
EQUS FNcopy2(&20E,savevec)
```

which would save the contents of the BRK vector at &20E-&20F into a location presumably defined elsewhere. Obviously we can restore it with:

```
EQUS FNcopy2(savevec,&20E)
```

Note in passing that we save the A register at the beginning of the macro (PHA), and restore it at the end (PLA). This is a very good idea, because then this macro can be used in any program at all with impunity. Structured programs in assembler follow exactly the same principles as in a higher-level language: to make every macro, subroutine or procedure completely self-contained and to make no assumptions about anything which happens outside itself.

Listing 1 contains a definition of a macro for a very common but rather tedious assembler operation: printing a string. Here we can use the value of the return string of the EQUS function to good effect and so call the macro with the string to be printed as its argument. Note that even though looping is performed there are no labels defined in the macro: these should not be included in macros since calling the macro more than once in the same program will cause confusion in the assembler between the differing instances of the label.

Don't be put off by the appearance of relative jumps ("P%+17") in line 1030 - the value of the offset from P% which is required is very easily calculated by using 'P%+0' and doing a test assembly with a listing. You can then calculate the correct value to be substituted from your listing. After a while you become proficient at byte-counting in assembler, and anyway it only needs to be done once when you create and test the macro. You can use it for evermore in your programs once created and tested!

HINTS SECTION

As mentioned in the first article in this series, we are starting a small hints section with each article. There are a number of small points which don't easily fit in anywhere else and so can be gathered together into one place. So here's this month's:

1. You can re-use label names as long as you are jumping backwards since the assembler always takes the last definition as the current one. This is particularly useful in loops where a small backward loop can always be called '.loop' without bothering to think of new names (as examples see the same articles as mentioned above).

2. When interfacing to Basic, you can pick up input values from the resident integer variables A% to Z% rather than through the complications of CALL parameters. A% is found at &404-407, B% at &408-40B etc. This is particularly handy in debugging and testing small modules.

   Thus, for example:

```
A%=N
CALL sort
...
.sort LDA &404 \ = A% MOD 256
```

Listing 1

```
  10 REM Practical Assembler
  20 REM Version B1.00
  30 REM Author  Bernard Hill
  40 REM BEEBUG  June 1990
  50 REM Program subject to copyright
  60 :
 100 DIM code 100
 110 x$="Example print macro"+CHR$13
 120 FOR opt=0 TO 3 STEP 3
 130 P%=code
 140 [ OPT opt
 150 EQUS FNprint(x$)
 160 EQUS FNprint("(C) BEEBUG 1990"+CHR
$13)
 170 RTS
 180 ]
 190 NEXT
 200 :
 210 CALL code
 220 END
 230 :
1000 DEF FNprint(a$)
1010 [ OPT opt:PHA:TXA:PHA
1020 LDX #0
1030 LDA P%+17,X
1040 JSR &FFE3
1050 INX:CPX #LENa$
1060 BNE P%-9
1070 PLA:TAX:PLA
1080 JMP P%+3+LENa$
1090 ]:=a$
```

Apricote Studios' Personal Accounts

Reviewed by Mike Williams

Product	**Personal Accounts**
Supplier	**Apricote Studios** **2 Puris Bridge Farm,** **Manea, Cambs PE15 0ND.** **Tel. (035 478) 432**
Price	**£14.95 inc.**

I must start this review by making a confession. Many years ago, or so it seems, I reviewed a number of home accounts packages for the BBC micro, and found nothing that appealed. Partly as a consequence, when I was writing the Filer database system which was published in BEEBUG in several instalments, one of the modules I included was for home banking. It has served me, and I hope others well, and I have now been using this regularly for about five years. Of course, it does all I want, and in a very simple fashion. Thus any commercial (and rival) package has much to live up to!

Apricote Studios' *Personal Accounts* is supplied on disc with a 20 page A5-sized manual. The software runs in mode 7, and despite the author's claims, I cannot approve of all the colour combinations (green text on a white background, for example). Mode 7 also limits descriptions of entries more than I would wish, and this field must contain both description and cheque number, if you feel that it is important to enter both (as I do if I am subsequently going to reconcile my computer record with the bank's statement). However, mode 7 makes the maximum amount of memory available which is important here as a complete file is loaded into memory for use, and then re-saved afterwards. This, of course, imposes a more modest upper limit on the total number of transactions that can be contained in one file, compared with one accessed directly from disc, but this is most unlikely to be a problem for this application.

Personal Accounts starts with a main menu from which files may also be loaded and saved (a demo file is supplied and used to good effect in the documentation by way of example). The most frequently used option is likely to be that for entering transactions.

Each transaction is dated, and this is streamlined to avoid unnecessarily repetitive input, and also requires code letters to classify the transaction according to the form of income or expenditure. The codes you choose to use are displayed below the transaction panel, and switch automatically between income and expenditure as required. Finally a transaction is marked with a status (paid, unpaid, reconciled or blank).

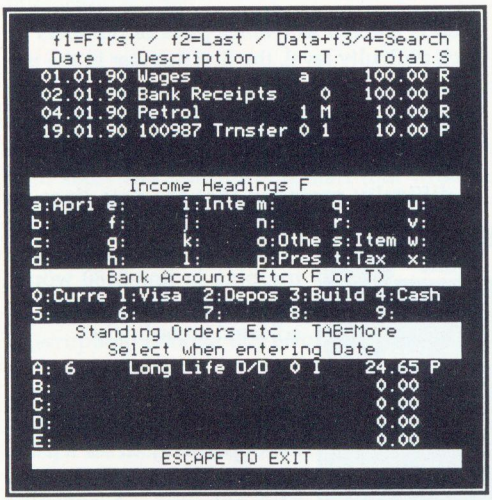
Personal Accounts is intended to cover all your income and expenditure, and thus entries can be attributed to one of ten accounts, of which your bank current account would be one. All entries can be easily edited, or deleted (by deleting the date for that entry). One excellent feature is that entries out of date order are immediately inserted into the correct position by date, thus always maintaining transactions in the correct order. Dates are also checked for validity.

A search facility enables you to search on any field in a transaction record. The search is case sensitive, which can be a hindrance if you are inconsistent in your descriptions, and the method of handling sub-string searches requires constant re-input of the search string. However, you will probably not need to make much use of this facility, and for what it does it is quite easy to use.

Handling standing orders and direct debits is one of the messier aspects of computerised home banking. *Personal Accounts* allows a maximum of 20, which from experience I would suggest is insufficient. For example, many regular payments involve one unique payment, and several equal ones. Over two years this can mean four separate entries in the standing orders list. In practice, with *Personal Accounts* it is preferable to have only one entry at a time for any one payment, so that additional editing and entry of standing orders is required.

Effectively, you create a separate list of standing orders/direct debit payments. You then *manually* insert appropriately dated copies of these into your transaction records as and when required. In my view this is quite unacceptable - one of the major reasons I use a

computer-based home accounts system is so that once I have told the computer the details of all my standing orders and direct debits, *it* will remember when to include them in a list of transactions. Admittedly, the software's author addresses this issue quite forcibly in his documentation, but I have to say that in my view his reservations can be overcome, and this has been proven to my satisfaction over the past five years.



The main data entry screen

Other options in the main menu allow you to enter and edit income and payment headings, and standing order/direct debit details. Another option summarises the different accounts you set up (current, building society, Access, Visa, cash etc.) showing current balances, which is quite handy. This is the screen where you also specify opening balances.

The remaining main menu option concerns the creation of reports and statements. All reports are displayed on the screen, and may be selectively sent to a printer. Reports all use an 80 character line, but the screen display is still limited to 40 characters. The result is most confusing, and tends to give an amateurish feel to the software. It also means that no more than three transactions are visible on screen at any one time. In reports, all codes are replaced by the correct descriptions, but the limits of the 40 column entry mode are then readily apparent. The screen display also scrolls straight through to the end unless you manage to press the space bar at the right moment to stall this, a less than satisfactory method in practice. You don't always want to print a report just to check a small detail (assuming you have a printer anyway), and a paged rather than scrolling display would be more effective here.

Various search criteria enable reports to select which accounts, payment or income headings are to be included, and what information is to be sent to a printer. The report option also includes the facility to carry current balances and all other settings forward to a new file (for the start of a new year for example).

The documentation, though brief, is actually quite good. The descriptions are clear and down to earth, and guide you easily through all the options. There are some grammatical errors which should have been spotted, but these are few. The only weakness is the lack of any technical detail, the maximum possible number of transactions per file, for example.

CONCLUSIONS

Many readers may feel I have been too hard on this package, but I explained my credentials at the outset. I need a system which is simple and straightforward in use, and which does not rely on my memory (other than to enter transactions) nor on my ability (or lack of it) at basic arithmetic. *Personal Accounts* does not fulfil all my requirements, particularly in its poor handling of standing orders/direct debits, and the limitations of a 40 column screen for entry and reporting. However, it is easy to use, and what it does it does well.

I looked forward to this review with keen anticipation. I am disappointed that I cannot report more favourably. The software is by no means expensive, and undoubtedly has quite a lot going for it, but it does not suit me. And after eight years in the life of the BBC micro I also believe we have a right to expect something a little more sophisticated, particularly when you bear in mind that the 3.5" ADFS version of *Personal Accounts* will, and is intended to be, run on an Archimedes, as well as the Master Compact I used for this review.

However, the author says he guarantees a refund of your money if you are not fully satisfied, so you have little to lose.

Programs for Education

Decimal Squeeze

This month's educational program tests children's understanding of decimal numbers, but will likely challenge adults too. Gordon Moxon explains.

It is not uncommon for children of even twelve or thirteen to have no real understanding of decimal numbers, especially of place value after the decimal point. This may be true despite the four rules - addition, subtraction, multiplication and division - being handled competently. It is at a later stage that problems arise.

The program presented here is designed to test a pupil's ability to cope with decimals in a challenging way and also, with the help of teachers or parents, to indicate problem areas and their solutions.

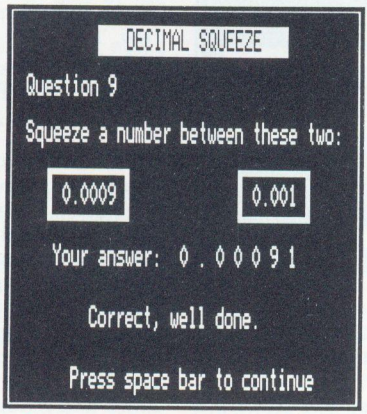
USING THE PROGRAM

You will first need to type in and save the program as listed, paying particular attention to the DATA lines. The program should not be renumbered as the RESTORE in line 2190 depends upon the DATA starting at lines 5000, 6000 and so on.

When first run, instructions are offered which invite pupils to squeeze a decimal number (not fraction) between two given numbers. Occasionally, children have thought that they have to provide a number exactly half way between the two numbers displayed and, although this is rare enough not to justify another screen of instructions, it does need watching out for.

A choice of difficulty is then offered. The easy level starts with two numbers which call for a simple integer answer and leads up to the two given numbers being consecutive whole numbers requiring a decimal answer for the first time. The highest level of difficulty, on the other hand, would demand a full understanding of place value and careful thought to master. It would be a fairly easy task to alter the data so as to tailor the standard of questions to suit the attainment level of pupils.

Data is entered as twenty pairs of numbers for each difficulty level, the first number in the pair being the lower number each time. Data for level one must start at line 5000, for level two at 6000 and so on. PROCchoice (lines 2090 to 2200) may be altered to allow for more or less levels of difficulty provided that the data lines for each additional level are numbered correctly. The error trapping in line 2180 would, of course, also have to be adjusted accordingly.



The squeeze program in operation

After twenty questions (set in line 1300) the user is offered the opportunity to see a list of any questions he/she may have got wrong. No stress is placed upon what is wrong for the sake of it. Rather a record is kept to enable pupils and teachers or parents to discuss the cause(s) of mistakes at the end of each set of questions. A record is also kept of each attempt, including level of difficulty, score out of twenty and time taken.

When children first learn about numbers, they quickly (and correctly) appreciate that the larger the number of digits used the greater the value of the number. Hence, for example, 93231 is bigger than 932. If later, place value is not fully understood, decimal numbers with many digits are often regarded as being large numbers and this is likely to be a frequent source of error. For example, 8.84 will be considered smaller than 8.83501 and even 0.009766 would be thought a large number! It is surprising how children who are able

mathematicians find the tasks set by this program difficult until their woolly thinking is clarified.

In over eighteen months of extensive classroom use the program has proved most useful in identifying and solving pupils' misconceptions about decimal numbers. It will not do this unaided, however, and pupil - teacher/parent interaction is vital.

PROGRAM NOTES

To assist entering the printed listing into the computer error trapping is include at line 100 and lines 190 to 220. As it stands, Escape will then abort and return to the start (but without repeating the instructions) while pressing Shift-Escape will immediately terminate the program.

If you want to make the program proof against interruption, remove this error trapping (when the program has been fully tested), and also remove the word REM at the start of lines 1190, 1210 and 1250. When this is done, pressing Escape or Break merely re-runs the program - the only way to terminate execution is to press Ctrl-Break.

```
   10 REM Program Decimal Squeeze
   20 REM Version B1.3
   30 REM Author  Gordon Moxon
   40 REM BEEBUG  June 1990
   50 REM Program subject to copyright
   60 :
  100 MODE7:ON ERROR GOTO190
  110 PROCinit:PROCinstr
  120 FOR go=1 TO 15
  130 PROCchoice
  140 PROCquestions
  150 PROCscore:IF go=15 THEN PROCend EL
SE IF exit% go=15
  160 NEXT go:MODE7:OSCLI("FX4"):*FX229
  170 END
  180 :
  190 MODE7
  200 IF ERR=17 AND NOT INKEY-1 THEN GOT
O120
  210 REPORT:PRINT" at line ";ERL
  220 END
  230 :
```

```
 1000 DEF PROCbox(X$,X%,Y%)
 1010 Z%=LEN(X$)+2
 1020 a$=CHR$150+CHR$104+STRING$(Z%,CHR$
44)+CHR$52
 1030 b$=CHR$150+CHR$106+CHR$135+X$+CHR$
150+CHR$53
 1040 c$=CHR$150+CHR$42+STRING$(Z%,CHR$4
4)+CHR$37
 1050 PRINTTAB(X%,Y%)a$
 1060 PROCdbl(b$,X%-1,Y%+1,32):PRINTTAB(
X%,Y%+3)c$
 1070 ENDPROC
 1080 :
 1090 DEF PROCdbl(X$,X1%,Y1%,C%)
 1100 PRINTTAB(X1%-1,Y1%)CHR$141;CHR$C%;
X$;TAB(X1%-1,Y1%+1)CHR$141;CHR$C%;X$;:VD
U11
 1110 ENDPROC
 1120 :
 1130 DEF PROCtitle
 1140 CLS
 1150 PROCdbl(CHR$150+CHR$157+CHR$132+CH
R$141+"DECIMAL SQUEEZE   "+CHR$156,8,1,3
2)
 1160 ENDPROC
 1170 :
 1180 DEF PROCinit
 1190 REM ON ERROR RUN
 1200 *FX4,2
 1210 REM *FX229,1
 1220 VDU23,1,0;0;0;0
 1230 DIM A$(20),B$(20),C$(20),E(20),AW$
(20),BW$(20),CW$(20)
 1240 DIMS(15),F(15),min$(15),SEC$(15)
 1250 REM *KEY10 O.|MRUN|M
 1260 ENDPROC
 1270 :
 1280 DEF PROCquestions
 1290 S=0:W=0:TIME=0
 1300 FOR X=1 TO 20
 1310 PROCtitle
 1320 XX$=STR$(X)
 1330 READ A$(X),B$(X)
 1340 IF LEN(A$(X))>LEN(B$(X)) E(X)=LEN(
A$(X))+2 ELSE E(X)=LEN(B$(X))+2
 1350 PROCdbl("Question "+XX$,1,4,130)
 1360 PROCdbl("Squeeze a number between
these two:",1,7,130)
 1370 PROCbox(A$(X),3,10)
 1380 PROCbox(B$(X),24,10)
 1390 PROCdbl("Your answer: ",4,15,130)
 1400 PROCinput
 1410 IF VAL(C$(X))<VAL(B$(X)) AND VAL(C
$(X))>VAL(A$(X)) PROCright ELSE PROCwron
g
```

```
 1420 NEXT X
 1430 ENDPROC
 1440 :
 1450 DEF PROCright
 1460 PROCdbl("Correct, well done.",8,19
,131)
 1470 PROCspace
 1480 S=S+1
 1490 ENDPROC
 1500 :
 1510 DEF PROCwrong
 1520 W=W+1
 1530 start$="No, "+C$(X)+" is equal to
"
 1540 IF VAL(C$(X))> VAL(B$(X)) PROCdbl(
"No, "+C$(X)+" is greater than "+B$(X),1
,19,129) ELSE IF VAL(C$(X))<VAL(A$(X)) P
ROCdbl("No, "+C$(X)+" is less than "+A$(
X),1,19,129)
 1550 IF VAL(C$(X))=VAL(A$(X)) PROCdbl(s
tart$+A$(X),1,19,129)
 1560 IF VAL(C$(X))=VAL(B$(X)) PROCdbl(s
tart$+B$(X),1,19,129)
 1570 PROCerrors
 1580 PROCspace
 1590 ENDPROC
 1600 :
 1610 DEF PROCinput
 1620 E=E(X):C$=""
 1630 REPEAT
 1640 P$=GET$:P=ASC(P$)
 1650 IFP=46 AND INSTR(C$,".") VDU7:GOTO
 1700
 1660 IF (P>47 AND P<58 OR P=46) AND LEN
(C$)<=E C$=C$+P$:VDU141,P,8,8,10,141,P,1
1:GOTO 1700
 1670 C=LEN(C$)
 1680 IF P=127 AND C=0 VDU7:GOTO 1700
 1690 IF P=127 C$=LEFT$(C$,C-1):C=C-1:VD
U127,9,10,127,8,11
 1700 UNTIL P=13 AND LEN(C$)>0
 1710 C$(X)=C$
 1720 ENDPROC
 1730 :
 1740 DEF PROCscore
 1750 PROCtitle
 1760 S$=STR$(S):T=TIME/100
 1770 MIN=INT(T/60):MIN$=STR$(MIN)
 1780 SEC=T MOD 60:SEC$=STR$(SEC)
 1790 IF MIN=1 MIN$=MIN$+" minute " ELSE
 MIN$=MIN$+" minutes "
 1800 PROCdbl("You got "+S$+" right out
of 20",7,7,130)
 1810 PROCdbl("in "+MIN$+"and "+SEC$+" s
econds",6,9,130)
 1820 IF S>18 AND FF<>52 PROCdbl("Excell
ent, now try the next level",2,11,130)
 1830 IF S<15 PROCdbl("Now try the same
level again",6,11,130)
 1840 PROCdbl("Do you want to check your
 errors",3,14,129)
 1850 PROCdbl("(Y/N)?",17,16,129)
 1860 REPEAT:W$=CHR$(GET AND &DF):UNTIL
W$="Y" OR W$="N"
 1870 IF W$="Y" PROCshowerrors
 1880 PROCrecord
 1890 IF go<15 PROCagain
 1900 ENDPROC
 1910 :
 1920 DEF PROCinstr
 1930 PROCtitle
 1940 PROCdbl("Do you want instructions
(Y/N)?",4,9,131)
 1950 REPEAT:W$=CHR$(GET AND &DF):UNTIL
W$="Y" OR W$="N"
 1960 IF W$="N" ENDPROC
 1970 PROCtitle
 1980 PROCdbl("This program tests your k
nowledge of",1,5,134)
 1990 PROCdbl("decimal numbers.",1,7,134
)
 2000 PROCdbl("You will be given two num
bers and you",1,9,134)
 2010 PROCdbl("have to type in a number
that is",1,11,134)
 2020 PROCdbl("between the two numbers a
nd not equal",1,13,134)
 2030 PROCdbl("to either of them. You mu
st put in a",1,15,134)
 2040 PROCdbl("decimal number and not a
fraction.",1,17,134)
 2050 PROCdbl("There are 20 questions al
together.",3,20,130)
 2060 PROCspace
 2070 ENDPROC
 2080 :
 2090 DEF PROCchoice
 2100 FOR K=1 TO 20:AW$(K)="0":NEXT
 2110 PROCtitle
 2120 PROCdbl("Which level of difficulty
 do you want?",1,5,131)
 2130 PROCdbl("1."+CHR$134+"Easy",10,8,1
30)
 2140 PROCdbl("2."+CHR$134+"Medium",10,1
0,130)
 2150 PROCdbl("3."+CHR$134+"Difficult",1
0,12,130)
 2160 PROCdbl("4."+CHR$134+"Very difficu
lt",10,14,130)
 2170 PROCdbl("Type the number required"
```

```
,8,18,129)
 2180 REPEAT:F=GET:UNTIL F>48 AND F<53:F
F=F:F=F-44
 2190 RESTORE F*1000
 2200 ENDPROC
 2210 :
 2220 DEF PROCspace
 2230 PROCdbl("Press space bar to contin
ue",6,23,129)
 2240 REPEAT:UNTIL GET=32
 2250 ENDPROC
 2260 :
 2270 DEF PROCagain
 2280 PROCtitle:exit%=FALSE
 2290 PROCdbl("Do you want another go (Y
/N)?",5,12,129)
 2300 REPEAT:V$=CHR$(GET AND &DF):UNTIL
V$="Y" OR V$="N"
 2310 IF V$="N" exit%=TRUE
 2320 ENDPROC
 2330 :
 2340 DEF PROCerrors
 2350 AW$(X)=A$(X):BW$(X)=B$(X):CW$(X)=C
$(X)
 2360 ENDPROC
 2370 :
 2380 DEF PROCshowerrors
 2390 PROCtitle
 2400 IF W=0 PROCdbl("You made no mistak
es!",9,10,130):PROCspace:ENDPROC
 2410 IF W>14 PROCdbl("You are either on
 the wrong level",1,10,129)
 2420 IF W>14 PROCdbl("or you need help
with decimals",3,12,129):PROCspace:CLS
 2430 PROCdbl("1st Number    2nd Number
 Answer",1,4,130)
 2440 PROCdbl(STRING$(35,"_"),1,6,131)
 2450 FOR J=0 TO 20
 2460 IF AW$(J)<>"0"PRINTTAB(4)CHR$134;A
W$(J);TAB(19)BW$(J);TAB(28)CHR$129;CW$(J
)
 2470 NEXT J
 2480 PROCspace
 2490 ENDPROC
 2500 :
 2510 DEF PROCrecord
 2520 CLS
 2530 PROCdbl(CHR$150+CHR$157+CHR$132+CH
R$141+"DECIMAL SQUEEZE RECORD  "+CHR$156
,5,1,32)
 2540 S(go)=S:F(go)=FF:min$(go)=STR$(MIN
):SEC$(go)=SEC$
 2550 IF LEN(SEC$(go))=1 SEC$(go)="0"+SE
C$(go)
 2560 PROCdbl("Go      Level      Score
 Time",3,3,130)
 2570 PROCdbl(STRING$(35,"_"),1,5,131)
 2580 FOR G=0 TO go
 2590 PRINTTAB(3)CHR$134;G;TAB(13)CHR$F(
G);TAB(23)S(G);TAB(31)min$(G);":";SEC$(G
)
 2610 NEXT
 2620 PROCspace
 2630 ENDPROC
 2640 :
 2650 DEF PROCend
 2660 PROCtitle
 2670 PROCdbl("You have completed 15 att
empts.",3,10,131)
 2680 PROCdbl("Re-run the program to try
 again.",3,14,131)
 2690 PROCspace
 2700 ENDPROC
 2710 :
 5000 DATA 234,245,46,56,6998,7003,345,3
46,21,210
 5010 DATA4.5,4.6,0.007,0.009,78,78.1,99
.9,100.1,6998,6999
 5020 DATA5.5,5.6,0,0.1,19,19.1,24,25,4.
75,4.76
 5030 DATA5.5,6.5,300,300.2,3.9,4,76,76.
2,0.01,0.08
 5040 :
 6000 DATA4.5,4.6,0.007,0.009,78,78.1,99
.9,100.1,6998,6999
 6010 DATA0.58,0.6,27.6,27.7,3.3,3.4,2.9
,3,3.8,3.82
 6020 DATA0.1,0.2,7.09,7.11,7.38,7.39,0.
34,0.35,1.45,1.46
 6030 DATA1.029,1.03,4.6,4.66,7.23,7.233
,0.01,0.02,18.117,18.118
 6040 :
 7000 DATA0.72,0.73,1000,1000.1,0.29,0.3
,0.299,0.3,0.36,0.63
 7010 DATA8.72,8.722,99,99.09,3.8,3.801,
0.0009,0.001,0,0.0001
 7020 DATA1987,1988,0.058,0.0582,11.011,
11.0121,19,19.7,0.804,0.80405
 7030 DATA7000,7000.1,9.204,9.24,9.201,9
.202,180,182,2.70,2.701
 7040 :
 8000 DATA 266.1,266.11,57.88,57.98,0.07
,0.0705,7,7.0001,1.7299,1.73
 8010 DATA0.0538,0.05382,27.08,27.8,42.0
01,42.0011,0,0.0001,3.57,3.9
 8020 DATA99.99,99.999,8.008,8.080,30,30
.001,1,1.01,0.05,0.06
 8030 DATA67.1,67.101,8642.3,8652.3,6.84
,6.84001,26.77,26.78,7.63,7.63003
```

A Menu Routine

by Paul Pibworth

Most of us are accustomed to menus of one sort or another. Primarily, they can be used in programs to give the user an element of choice. If the user is the programmer, then the menu system will always seem easy to handle. The crunch comes when others have to use the same program. If there are many options in a menu, the user, especially if unsympathetic, tends to develop a glazed expression, as though mesmerised by the screen. It is just an extension of the phenomenon which afflicts many of us (well at least me) when presented with decision making and information overload. The answer is to simplify the process, by means of structured questions, or choices. This technique is used on some forms, which contain instructions such as "If NO go to question 8".

The program presented here is a menu system, which implements a more structured approach to menus as outlined above. It is designed to be adapted for your own use, and to illustrate some of the programming techniques used for the benefit of First Course readers. If it is typed in and run, it will give a demonstration of how the menu system works. It can then be saved for incorporation into your own programs later. In order to use it, you need to know what you want to do, and a little of what happens in the program.

In principle, the menu system is organised so that the number of visible choices is kept to a sensible and reasonable limit at any one time. Choosing any option may lead to a further sub-menu and the process can be repeated with further levels of sub-menu if required. This *hierarchical* menu system ensures that only a limited number of choices are presented on screen together.

There needs to be a balance between offering a choice of, say, 1 from 2, and, at the other extreme say, of 1 from 10. I have chosen 1 from a maximum of 5 (but always making the fifth one an exit route of some kind). Once you get to know the program, it can be changed to allow a choice of 1 from 3, or 1 from 5 (or whatever else you prefer). This would need changes being made to the procedure PROCmenu, together with appropriate changes made in the number of parameters sent and received, as you will see in due course.

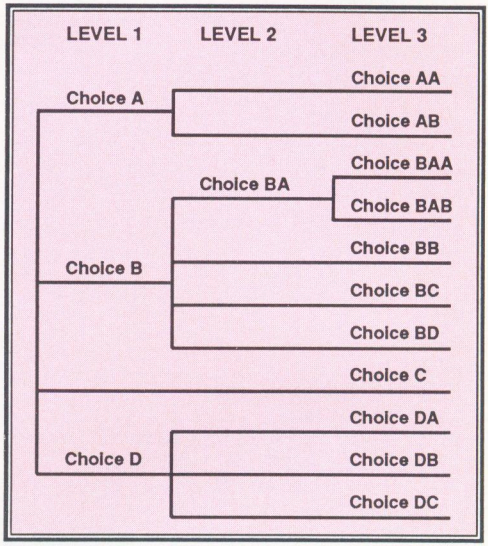


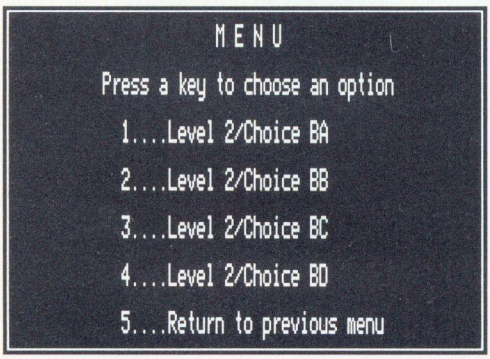
Figure 1

However, your first need is to draft a plan such as figure 1. The system can be designed entirely to suit your own requirements, and is quite versatile. You will see in my example that choice C leads to no further choices, yet choice B leads to a further 4 choices, one of which leads to a third sub-menu. The text for each menu option is placed in a DATA statement, and my (somewhat artificial) choices are listed from line 600 onwards.

The program works by reading the text of the choices (i.e. the options as they will be displayed on the screen for each menu), from the DATA statements into a string array called *menu$*. This is accomplished in lines 100 to 160. By subsequently including the array position as a parameter in the procedure PROCmenu, the appropriate text is displayed.

The program has been written for mode 7, so the string variables in the array contain a colour code, and code 141 to give double height characters. The colour code has been included as the first value of each DATA statement. This and the '141' could be omitted and the program adapted for other modes if you wished. In addition, the first DATA statement is used to create a completely blank menu entry when required, and the next two entries are likely to be standard requirements for all menu systems based on this program. From then on, REM statements have been included with the data at intervals, as a reminder, to indicate the array positions of the following option choices.

A menu (or sub-menu) is displayed on the screen using the procedure PROCmenu. This has five parameters being the array references of the five options to be displayed. Usually, the fifth option will be 'Return to previous menu' or 'Exit system' (the latter in the top-level menu - i.e. level 1 - only). The blank option, entry 0 in the array, may be used whenever a menu is to contain less than five options.



Second level menu display

The menu demo begins at line 180 by clearing the screen and displaying a menu. The one shown is determined by the value of Z% which is initially set to FALSE (value 0) at line 190. Thus the top-level menu is specified in line 270 using the text from menu$ in positions 3, 4, 5, 6, and 7.

To complement PROCmenu, which displays a menu of specified options on the screen, the function FNmenu returns a value indicating which choice has been made, and this is assigned (suitably modified) to Z% to determine the next menu (or sub-menu) to be shown. The parameter used with FNmenu is a string indicating which options are valid. Thus the top-level menu specified in line 270 allows five options (12345), but the sub-menu for the first of these (see figure 1 again) allows only three (125). The whole of the menu system is contained in a REPEAT-UNTIL loop (lines 200 to 450).

Study lines 270 to 310. The values of Z% from the first choice must be from 100 to 500 (hence the multiplier of 100). If Z% is 100, then line 280 is invoked, so that Z% then leaves with a value of 110, 120, or 150. A value of 200 would become 210, 220, or 250, which in turn would modify 210 to 211, 212, or 215, whereas 300 would not be subjected to further change. The order of these statements is critical and follows the values for Z% - the 100s first, then the 200s and so on. A value of 500 indicates an exit from the menu systems altogether, assuming that the level 1 menu includes this as the fifth option.

You need to know the final values of Z% to use in the second half of the loop, i.e. lines 330 to 440. This is where the action for each of the eleven choices shown in figure 1 is finally determined. In the demo, each choice merely displays the name of that option, but in practice each would probably call a different procedure to perform the appropriate operation.

I included PROCpause in the program for effect, but it is not essential. By changing the value of the multiplying factor in line 1190, you can slow down or speed up the rate of appearance of the options on the screen.

Finally, to summarise the whole system, start by constructing a plan, as in figure 1, and then write your "choices" on the lines of the diagram, before transferring them to the DATA lines. Modify the parameters used by PROCmenu and FNmenu, and then try a run, before you add your own procedures in lines 280 and following. Because, as it stands, each choice calls the same procedure, the text of your choice will be displayed. The value of Z% is also displayed on the screen, and indicates the pathway followed by the user; e.g. 212 shows the choices 2, then 1, then 2. As soon as it is up and working, replace my PROCone by your own procedures, and you won't look back.

```
   10 REM Program FCMenu
   20 REM Version B2.2
   30 REM Author  Paul Pibworth
   40 REM BEEBUG  June 1990
   50 REM Program subject to copyright
   60 :
  100 d$=CHR$141
  110 DIMmenu$(20)
  120 RESTORE
  130 FOR I%=0 TO 18
  140 READcode%,menu$
  150 menu$(I%)=d$+CHR$code%+menu$
  160 NEXT
  170 :
  180 MODE7
  190 Z%=FALSE
  200 REPEAT
  210 CLS
  220 PRINTTAB(12,1)CHR$131;CHR$141;"M E
 N U"
  230 PRINTTAB(12,2)CHR$131;CHR$141;"M E
 N U"
  240 PRINTTAB(0,4);menu$(1);TAB(0,5);me
nu$(1)
  250 VDU28,0,24,39,6
  260 IF Z%=215 Z%=200 ELSE Z%=FALSE
  270 IF Z%=FALSE PROCmenu(3,4,5,6,7):Z%
=FNmenu("12345")*100
  280 IF Z%=100 PROCmenu(8,9,0,0,2):Z%=F
Nmenu("125")*10+100
  290 IF Z%=200 PROCmenu(10,11,12,13,2):
Z%=FNmenu("12345")*10+200
  300 IF Z%=210 PROCmenu(17,18,0,0,2):Z%
=FNmenu("125")+210
  310 IF Z%=400 PROCmenu(14,15,16,0,2):Z
%=FNmenu("1235")*10+400
  320 VDU26,12
  330 IF Z%=110 PROCone(menu$(8))
  340 IF Z%=120 PROCone(menu$(9))
  350 IF Z%=211 PROCone(menu$(17))
  360 IF Z%=212 PROCone(menu$(18))
  370 IF Z%=220 PROCone(menu$(11))
  380 IF Z%=230 PROCone(menu$(12))
  390 IF Z%=240 PROCone(menu$(13))
  400 IF Z%=300 PROCone(menu$(5))
  410 IF Z%=410 PROCone(menu$(14))
  420 IF Z%=420 PROCone(menu$(15))
  430 IF Z%=430 PROCone(menu$(16))
  440 IF Z%=TRUE THEN Z%=FNend
  450 UNTIL Z%=500
  460 CLS
  470 PRINTTAB(15,10)"END"
  480 END
  490 :
  600 DATA 0,""
  610 DATA 130,Press a key to choose an
option
  620 DATA 129,5....Return to previous m
enu
  630 REM array 3-7
  640 DATA 131,1....Level 1/Choice A
  650 DATA 131,2....Level 1/Choice B
  660 DATA 131,3....Level 1/Choice C
  670 DATA 131,4....Level 1/Choice D
  680 DATA 129,5....Exit System
  690 :
  700 REM array 8-9
  710 DATA 131,1....Level 2/Choice AA
  720 DATA 131,2....Level 2/Choice AB
  730 :
  740 REM array 10-13
  750 DATA 131,1....Level 2/Choice BA
  760 DATA 131,2....Level 2/Choice BB
  770 DATA 131,3....Level 2/Choice BC
  780 DATA 131,4....Level 2/Choice BD
  790 :
  800 REM array 14-16
  810 DATA 131,1....Level 2/Choice DA
  820 DATA 131,2....Level 2/Choice DB
  830 DATA 131,3....Level 2/Choice DC
  840 :
  850 REM array 17-18
  860 DATA 131,1....Level 3/Choice BAA
  870 DATA 131,2....Level 3/Choice BAB
  880 :
 1000 DEF PROCmenu(a%,b%,c%,d%,e%)
 1010 CLS
 1020 PRINTTAB(2,1);menu$(a%)
 1030 PRINTTAB(2,2);menu$(a%)
 1040 PROCpause(1)
 1050 PRINTTAB(2,4);menu$(b%)
 1060 PRINTTAB(2,5);menu$(b%)
 1070 PROCpause(1)
 1080 PRINTTAB(2,7);menu$(c%)
 1090 PRINTTAB(2,8);menu$(c%)
 1100 PROCpause(1)
 1110 PRINTTAB(2,10);menu$(d%)
 1120 PRINTTAB(2,11);menu$(d%)
 1130 PROCpause(1)
 1140 PRINTTAB(2,13);menu$(e%)
 1150 PRINTTAB(2,14);menu$(e%)
 1160 ENDPROC
 1170 :
 1180 DEF PROCpause(t%)
 1190 T=INKEY(t%*10)
 1200 ENDPROC
 1210 :
 1220 DEF FNmenu(a$)
 1230 REPEAT:Z$=GET$:UNTILINSTR(a$,Z$)>0
 1240 =VAL Z$
 1250 :
 1260 DEF FNend
 1270 CLS
 1280 PRINTTAB(10,10);CHR$131;"You are a
bout to quit"
 1290 PRINTTAB(11,12);CHR$130;"Press Y t
o confirm"
 1300 C$=INKEY$(500)
 1310 IF C$="Y" OR C$="y" THEN =TRUE
 1320 =FALSE
 1330 :
 1340 DEF PROCone(a$)
 1350 CLS
 1360 PRINTTAB(8,8);a$
 1370 PRINTTAB(8,9);a$
 1380 PRINT SPC17"Z%=";Z%
 1390 PRINT SPC14'"Press a key"
 1400 T=GET
 1410 ENDPROC
```

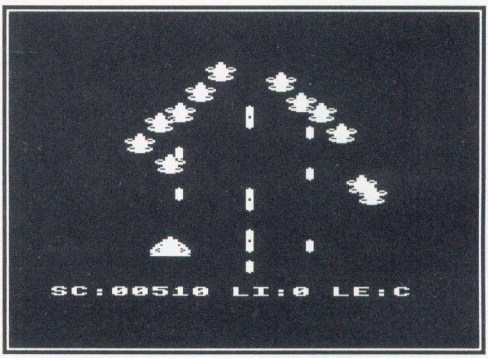
Designer Shoot-'Em-Up (Part 2)

Alan Wrigley concludes his description of Al Harwood's designer game routines.

In last month's article, we showed you how to define the sprites for your game, and listed the source program for the machine code routines. To complete the picture, you need to create the screens for the game, and this is what we shall concentrate on this month.

ENEMY MOVEMENTS

The first stage in defining the screens is to create an ASCII text file which will contain instructions about screen colours, which objects are to start where, and how they will move. The first part of this file will consist of a number of one-line instructions which specify the movements. This is followed by a further section for each screen, specifying the screen colours, how many enemy sprites, their starting positions on the screen, and which movement instruction to start from. Thus the instruction list in the first part of the file can contain movement patterns for any number of sprites, whose starting positions can be specified later.



A screen from the sample game given in figure 1

Movement instructions are taken from the following list:

skip - no function; used to slow down movement.
left - moves enemy 1 unit left.
right - moves enemy 1 unit right.
up - moves enemy 1 unit up.
down - moves enemy 1 unit down.
explode - enemy is killed.
man - moves enemy 1 unit towards player's position.
bom - enemy drops a bomb.
>n - cycles back n instructions in the program (where n<128).

All these instructions except the last can be abbreviated to the first letter (note that it must be lower case). So a sample program might be:

```
l
l
l
r
r
r
>6
```

which would simply make three moves left then three moves right in an infinite loop. Much more complex movements can of course be built up. For example, you could move an enemy sprite in a zig-zag pattern, dropping bombs and speeding up as it goes, then finally diving straight for the player.

Remember that all these instructions must occupy a line each, with no blank lines in between. After the movement patterns have been defined, however, you *must* leave a blank line before defining the rest of the screen information.

SCREEN DISPLAY

The next part of the file should be repeated for each screen you wish to include in your game. The instructions this time (again one per line) are:

1. *screen colours* (x,y,z)
2. *sprite number* (from your sprite file)
3. *x position* (0-39)
4. *y position* (0-30)
5. *map position*

2-5 repeated for each sprite that will appear on this screen.

The screen colours are the standard BBC screen colours in the range 1-15. Three numbers must be specified, with commas between them, e.g. *4,5,6*. The sprite number is the same number as that used in the sprite file you defined last month. The x and y positions specify the starting position on the screen, and the map position refers to the instruction number at which to start, from the list of movement instructions in the first part of the file. Note that you must not allow sprites to move off the screen or you might get unpredictable effects.

So the second part of the file might look like this:

```
4,5,6
2
10
10
0
2
20
10
13
```

which defines screen colours 4,5 & 6; places sprite 2 at position 10,10 and starts its movement at instruction 0; and places a second copy of sprite 2 at position 20,10 and starts its movement from instruction 13. There must be a blank line between each screen definition, and the end of the file must be marked by two blank lines.

The text file can be prepared by any word processor which has the facility to save text as an ASCII file. A sample text file is shown in Figure 1. For reasons of space, the file has been compressed; normally each instruction would occupy one line. This game requires three enemy sprites to be defined (nos. 2,3, & 4).

Now type in listing 3 and save it as *ScrnAsm*. When the program is run, it will prompt you for the filename of your text file, and will convert the instructions into the data necessary to construct the screens. When finished, you will be prompted for a filename for the resulting screens file.

Finally, type in listing 4, replacing *DEMOscr* and *DEMOspr* in lines 1870 and 1880 with your own screen and sprite filenames. Save the program as *Loader*.

```
l l l u u u d r d r d r r r r u u u r u
r u r u d d d l l l d l d l d l >37 s s
s s s s s s s s s s s s m >2 r s s d s s r
s s d s s r s s d s s r s s d s s r s s
d s s r s s d s s r s s d m >1 l l l l l l
l l l l l l l l l l l l l l l r r r r r r
r r r r r r r r r r r r r r >41 s s s
s s s s s s s s s s s s s s s s s b d l
d l d l d l d l d l d l b l d l d l d l d l
d l d l d l d l d l m m m m m m m m m e s
s s s s s s s s s s s s s s s s s s s s b
d r d r d r d r d r d r b r d r d r d r
d r d r d r d r d r d r d m m m m m m m m
m e b l l l b r r r b r r r b l l l >16

123 2 10 10 0 2 20 10 0 2 10 20 0 2 20
20 0 2 15 15 0

234 3 10 6 0 3 20 6 0 2 5 2 38 2 10 2 38
2 20 2 38 2 25 2 38

167 2 5 6 53 2 10 6 53 2 10 6 53 2 15 6
53 2 20 6 53 3 30 15 95

345 3 20 6 135 3 20 6 138 3 20 6 142 3
20 6 146 3 20 6 150 3 20 6 154 3 20 6
197 3 20 6 201 3 20 6 205 3 20 6 209 3
20 6 213 3 20 9 217 3 20 9 221

345 4 30 8 95 4 10 8 115 3 15 16 261 3
25 16 261
```

Figure 1

PLAYING THE GAME

You should now have all the ingredients necessary to play your game. These are:

1. A machine code program called *Game*, which was assembled from listing 2 last month.
2. A screen file, as described above.
3. A sprite file, created with the sprite definer listed last month.
4. A Basic program called *Loader*.

To run the game, first set PAGE to &1900 (Master owners included), and then chain the loader program. This will load all the necessary files and start the game running. To play the game, use Z and X to move the player, and Return to fire a bomb. Have fun!

Listing 3

```
   10 REM Program ScrnAsm
   20 REM Version 1.0
   30 REM Author  Al Harwood
   40 REM BEEBUG  June 1990
   50 REM Program subject to copyright
   60 :
  100 MODE7:PROCinit:PROCassemble:END
  110 :
 1000 DEF PROCinit
 1010 ld=&3000:REM load address
 1020 le=&1000:REM length
 1030 DIM wd$(8),data le
 1040 FOR A%=data TO data+le STEP 4
 1050 !A%=0:NEXT
 1060 FOR A=0 TO 8:READ wd$(A):NEXT
 1070 PRINT'''CHR$131"PROGRAM ASSEMBLER"
'CHR$131"================"'''CHR$130"By
Al Harwood"'''CHR$130"BEEBUG June 1990"''
'CHR$134;
 1080 INPUT"Source code filename:"fn$
 1090 OSCLI("E."+fn$)
 1100 ENDPROC
 1110 :
 1120 DEF PROCassemble
 1130 n=21:REPEAT INPUT"P>"a$
 1140 q=-1:IF a$>""PROCassem1
 1150 UNTIL q=-1:IF a$>"" ENDPROC
 1160 s=1:REPEAT INPUT"C>"c$
 1170 IF c$>""PROCassem2
 1180 UNTIL c$=""
 1190 CLOSE#0:*FX15
 1200 INPUT"Enter filename:"fn$
 1210 OSCLI("S."+fn$+" "+STR$~data+"+"+S
TR$~le+" 0 "+STR$~ld)
 1220 ENDPROC
 1230 :
 1240 DEF PROCassem1
 1250 FOR a=0 TO 8
 1260 IF wd$(a)=LEFT$(a$,1)q=a:a=8
 1270 NEXT:IF q=-1 PRINT''CHR$131"No suc
h command":CLOSE#0:ENDPROC
 1280 data?n=q
 1290 IFq=8 data?n=128+VALMID$(a$,2)
 1300 n=n+1:ENDPROC
 1310 :
 1320 DEF PROCassem2
 1330 data?s=(n+ld)MOD256:s=s+1
 1340 data?s=(n+ld)DIV256:s=s+1
 1350 ?data=?data+1
 1360 FOR I%=n TO n+2:x%=INSTR(c$,",")
 1370 a$=LEFT$(c$,x%-1)
 1380 c$=MID$(c$,x%+1)
 1390 data?I%=VALa$:NEXT:n=n+3
 1400 REPEAT INPUT"S>"a
 1410 IF a THEN data?n=a:n=n+1:INPUT"X>"
d,"Y>"b,"M>"c:data?n=d:n=n+1:data?n=b:n=
n+1:data?n=(c+ld+20)MOD256:n=n+1:data?n=
(c+ld+20)DIV256:n=n+1
 1420 UNTIL a=0
 1430 data?n=255:n=n+1
 1440 ENDPROC
 1450 :
 1460 DATA s,l,r,u,d,e,m,b,>
```

Listing 4

```
   10 REM Program Loader
   20 REM Version 1.0
   30 REM Author  Al Harwood
   40 REM BEEBUG  June 1990
   50 REM Program subject to copyright
   60 :
  100 HIMEM=&3000:PROCload:PROCinit
  110 PROCenvelopes:REPEAT PROCtitles
  120 PROCgame:PROCcompleted
  130 PROChighscore:UNTIL FALSE
  140 :
 1000 DEF PROCinit
 1010 A%=0:code=&1200
 1020 s1$="00000":s2$=s1$:s3$=s1$
 1030 s4$=s1$:s5$=s1$:n1$="Al Harwood"
 1040 n2$=n1$:n3$=n1$:n4$=n1$
 1050 n5$=n1$:s$="on "
 1060 ENDPROC
 1070 :
 1080 DEF PROCtitles
 1090 VDU 22,4,23;8202;0;0;0;
 1100 REPEAT PROCtitle1
 1110 UNTIL I=32:ENDPROC
 1120 :
 1130 DEF PROCtitle1
 1140 VDU 12,19,1,0;0;
 1150 PRINT''''SPC9FNdoub("DESIGNER SHO
OT-'EM-UP")''''SPC13FNdoub("By Al Harwoo
d")'''''SPC14FNdoub("Press SPACE")
 1160 PRINT''SPC9FNdoub("or H to start o
n the")'''SPC5FNdoub("highest level reac
hed so far")
 1170 VDU 19,1,1;0;:*FX15
 1180 I=INKEY200:IF I=32 ENDPROC
 1190 IF I=72 I=32:code=&1202:ENDPROC
 1200 VDU 12,19,1,0;0;
 1210 PRINT''''SPC8FNdoub("In the game t
o move use:")'''SPC17FNdoub("Z and X")''
'SPC11FNdoub("Hit RETURN to fire")''''SP
C12FNdoub("The sound is")
 1220 COLOUR 0:COLOUR 129
 1230 PRINTTAB(25,14)FNdoub(s$)
 1240 COLOUR 1:COLOUR 128
 1250 PRINT''SPC6FNdoub("To change this
```

```
use the S key")''''SPC14FNdoub("Press SP
ACE")
 1260 VDU 19,1,6;0;:*FX15
 1270 REPEAT I=INKEY500
 1280 IF I=83 OR I=115 s$=FNonoff:COLOUR
 0:COLOUR 129:PRINTTAB(25,14)FNdoub(s$):
COLOUR 1:COLOUR 128
 1290 UNTIL I=32 OR (I<>83 AND I<>115)
 1300 IF I=32 ENDPROC
 1310 VDU 12,19,1,0;0;
 1320 PRINT'''''SPC14FNdoub("HIGH  SCORE
S")''''SPC6FNdoub("1. "+s1$+" _______ "+
n1$)'''SPC6FNdoub("2. "+s2$+" _______ "+
n2$)'''SPC6FNdoub("3. "+s3$+" _______ "+
n3$):*FX200,3
 1330 PRINT''SPC6FNdoub("4. "+s4$+" ____
___ "+n4$)'''SPC6FNdoub("5. "+s5$+" ____
___ "+n5$)''''SPC14FNdoub("Press SPACE")
___
 1340 VDU 19,1,3;0;
 1350 I=INKEY400:ENDPROC
 1360 :
 1370 DEF PROCgame
 1380 VDU 22,5,23;8202;0;0;0;
 1390 IF s$="on ":*FX210
 1400 IF s$="off":*FX210,1
 1410 CALL code:code=&1200
 1420 ENDPROC
 1430 :
 1440 DEF PROCcompleted
 1450 IF ?&8B=0 ENDPROC
 1460 VDU 22,4,23;8202;0;0;0;19,1,0;0;
 1470 PRINT''''''SPC12FNdoub("CONGRATULA
TIONS"):*FX15
 1480 C=0:REPEAT C=C+1:IF C=8 C=1
 1490 VDU 19,1,C;0;:UNTIL INKEY10>-1
 1500 ENDPROC
 1510 :
 1520 DEF PROChighscore
 1530 VDU 22,4,19,1,3;0;
 1540 IF A%<?&87 A%=?&87
 1550 sc$=CHR$?&70+CHR$?&71+CHR$?&72+CHR
$?&73+CHR$?&74:sc=VALsc$
 1560 IF sc>VALs1$ s5$=s4$:s4$=s3$:s3$=s
2$:s2$=s1$:s1$=sc$:n5$=n4$:n4$=n3$:n3$=n
2$:n2$=n1$:n1$=FNname:ENDPROC
 1570 IF sc>VALs2$ s5$=s4$:s4$=s3$:s3$=s
2$:s2$=sc$:n5$=n4$:n4$=n3$:n3$=n2$:n2$=F
Nname:ENDPROC
 1580 IF sc>VALs3$ s5$=s4$:s4$=s3$:s3$=s
c$:n5$=n4$:n4$=n3$:n3$=FNname:ENDPROC
 1590 IF sc>VALs4$ s5$=s4$:s4$=sc$:n5$=n
4$:n4$=FNname:ENDPROC
 1600 IF sc>VALs5$ s5$=sc$:n5$=FNname
 1610 ENDPROC
 1620 :
 1630 DEF FNdoub(d$):LOCAL A%
 1640 X%=&70:Y%=0:A%=10
 1650 FOR D%=1 TO LENd$
 1660 ?&70=ASCMID$(d$,D%,1):CALL &FFF1
 1670 VDU 23,159,?&71,?&71,?&72,?&72,?&7
3,?&73,?&74,?&74,159,10,8,23,159,?&75,?&
75,?&76,?&76,?&77,?&77,?&78,?&78,159,11
 1680 NEXT:=CHR$0
 1690 :
 1700 DEF FNname:CLS:PRINT''''FNdoub("Y
ou have a new high score")''''FNdoub("En
ter your name: ")CHR$10;:*FX15
 1710 n$="":REPEAT K=GET:IF K=13 n$=n$+S
TRING$(11-LENn$," ") ELSE IF K=127 AND L
ENn$>1 VDU 8,32,8,11,32,10,8:n$=LEFT$(n$
,LENn$-1) ELSE IF K>31 n$=n$+CHR$K:PRINT
;CHR$11FNdoub(CHR$K)CHR$10;
 1720 UNTIL LENn$=11:CLS:=n$
 1730 :
 1740 DEF FNonoff
 1750 IFs$="on ="off" ELSE ="on "
 1760 :
 1770 DEF PROCenvelopes
 1780 ENVELOPE 1,1,43,0,0,100,0,0,126,0,
0,-126,126,126
 1790 ENVELOPE 2,12,10,8,6,4,0,0,126,0,0
,-126,126,126
 1800 ENDPROC
 1810 :
 1820 DEF PROCload
 1830 REM   Replace the first with
 1840 REM   programs filename,
 1850 REM   and second with the
 1860 REM   sprites filename.
 1870 *L.DEMOscr
 1880 *L.DEMOspr
 1890 *L.GAME
 1900 ENDPROC
```

B

Points Arising....Points Arising....Points Arising....Points Arising....

HINT - FLAGGING EVENTS (Vol.8 No.8)

An accidental copying error led to the last line at the bottom left of this page being incorrect. It should read:

```
IF (M?A AND 2^B)=0 THEN M?A=M?A+2^B
```

Thanks to Mr.Burton-West of London for spotting this.

B

512 Forum

by Robin Burton

I seem to have had a number of letters over the last month or so from newer converts to the 512, so some second hand 512s have found good homes with Forum readers!

Some of these letters contained requests for more basic hints and tips. I said a couple of months ago, let me know what you want to see in the Forum and these readers have done just that. Because of that and also because it's a while since we did so, this month we'll take a look at a few basic DOS commands.

OUTSTANDING MATTERS

Before going on to DOS commands though, as promised I'll carry on with a bit more information about MS-DOS, supplied by my anonymous correspondent (I'll call him Bill for convenience).

As an addendum to last month's DOS versions, I have been told that Amstrad's version of MS-DOS doesn't respond to this treatment, but that release 2.2 of Tandon MS-DOS 3.2 definitely does. Again though I must warn you to check copyright agreements before you try it.

As I mentioned, this dodge leaves even less memory than normal in the 512, so it's no help for general software use. Also, because the operating system calls ultimately must still be serviced by DOS Plus, if a program doesn't run in the 512 under DOS Plus on its own this trick won't provide a magical cure. In other words whatever limitations DOS Plus normally imposes still lurk behind the MS-DOS shell.

What you may find though, is that programs you can't install or configure on the 512 because they check the version of DOS in use, may be fooled into letting you set them up for eventual use under DOS Plus. Most of the programs in this category will produce a clear message of 'Requires MS-DOS version x' or 'Incorrect version of DOS' or something of the sort.

However, not all programs will necessarily be this helpful, and I've found that sometimes the error message produced in DOS Plus has little or nothing to do with the real problem. It seems the authors simply didn't think of DOS Plus when they wrote their program and the error message can be a bit of a lottery. If you have a program that can't be installed under DOS Plus, even if you get a different message or even none at all, it's certainly worth trying to install it again this way (unless you've already tried configuring on a real PC and the resulting system still doesn't run).

For those without a Winchester, no doubt the majority of 512 users, another problem also occurs quite frequently when trying to install PC software. Some programs insist on being set up from drive C:. Why this is done I don't know. There's no logical reason, and an equally large proportion of PCs, again probably the majority, don't have a hard disc either.

There is, however, in MS-DOS a program called 'ASSIGN'. This program permits the user to nominate an existing drive to take on an 'alias' for another. In MS-DOS therefore, if you have only floppies you can, for example, pretend that drive B: is called drive C: to avoid this sort of problem. There are other ways to overcome this problem in the 512 directly in DOS Plus, as some readers will know, but those who don't will have to read Bernard Hill's reviews again to find out about them (see Essential Software reviews in Vol.8 No.5 and Vol.9 No.1).

The point is that if you have been able to load MS-DOS as described, ASSIGN can be used in the 512 for the purpose of software configuring. ASSIGN will not, however, run directly under DOS Plus.

Bill has configured Finesse version 1.1 using this technique (he hasn't yet tried version 2), but warns that although Finesse then runs under GEM 2 you'll have to delete the desktop accessories first to create enough free memory in an unexpanded 512. Even so, he warns that there's only enough free space to load the three smallest fonts (Swiss or Dutch), but apparently it's quite usable within this limitation.

Bill also says that it's not possible to use the bitstream software supplied with Finesse without a Winchester. The problem is that there's just not enough disc space, even with 800K floppies. He says, and I quote, "...it needs about a megabyte just to install, and a phenomenal amount to create fonts...".

Bill also tells me that GEM 3 can be used too, but it leaves even less free memory and so is even more restrictive than version 2. If memory isn't a problem you should replace the standard CGA driver with AcornBW.SYS from the 512'S GEM disc, and then the mouse can be used (presumably those with a Tull Mouse driver can use that instead - I haven't tried it myself, but GEM 3 is shown in Tull's manual as compatible).

DOS COMMANDS AGAIN

Now on to some of the basic commands in DOS, starting with common disc commands. These are, in the main, very easy to use and very flexible, much more so than in the BBC's filing systems, but I wonder how many of you have fully explored them. If you haven't you might be making things harder than they need to be.

Let's start with 'COPY'. Suppose that you wanted to copy all the files in the current directory in drive A: to drive B:. Assuming drive A: is the current drive, I wonder how many of you would enter:

```
COPY *.* B:
```

when

```
COPY A: B:
```

would do just as well. O.K., that only saves one character, but what about:

```
COPY . B:
```

You've all seen the single and double dots in a directory display. In case you didn't know the single dot means the current directory and the double dot means the parent of the current directory. If you thought they were just for decoration let me tell you they're not, they are very useful.

MOVING FILES

Suppose you're in a directory three levels down from the root and want to copy all 'EXE' files to the directory above the current one. Let's assume for simplicity that the directories are called 'DIR1', 'DIR2' and 'DIR3', but bear in mind that directory names are usually longer, with more chances for typing errors. Doing things the hard way you could enter the command as:

```
COPY \DIR1\DIR2\DIR3\*.EXE \DIR1\DIR2
```

With longer names and done this way, it becomes a very lengthy command. The first point of course, and it's one that you should all have spotted at once, is that the current directory name needn't be specified at all. If you don't supply a directory name the current one is used by default, which makes things a bit better, but the command is still rather clumsy in its revised form:

```
COPY *.EXE \DIR1\DIR2
```

when you could much more easily and simply used the double dots. In DOS these are the equivalent of '^' in ADFS, meaning the parent directory of the current one, so the command can be made very much shorter and simpler by entering:

```
COPY *.EXE ..
```

with very little chance of error.

So far I've used copies to the same disc for these examples, but if you actually want to move the files to a different directory on the same disc, rather than to duplicate them, this isn't the right way to do things. One reason is that you need to delete the original directory contents

afterwards, but worse is that during the copy you need twice the disc space, and copying takes a long time.

Far better is to rename files, when DOS only needs to move the directory entries leaving the original files intact. Clearly it's very much faster, it requires no extra disc space because only one copy of the file ever exists, and there's no need to delete old versions afterwards because there aren't any. Our first command therefore becomes:

```
REN *.EXE ..\
```

DIFFERENT DRIVES

Next, there are a few points to make about copying between different drives. No-one was surprised when I mentioned that the current directory can always be defaulted. You probably use the fact without even thinking about it, especially on floppies when the number and depth of paths tends to limit complexity. Even so, when you issue a command like:

```
COPY A: B:
```

remember you're actually telling DOS to copy from the current directory in drive A: to the current directory in drive B:, because you've defaulted both of them. Yes, all very simple I agree, but had you realised that the single and double dots can also be used between drives?

For example, to copy all the files from the current directory in the current drive (A:) to the parent of the current directory in drive B: the command can be simplified to:

```
COPY . B:..
```

which is just about as short as you could hope for.

One criticism of these short-cuts is that they're rather cryptic, and it's sometimes easy to overlook just what's going on. This is quite true, but it's always easy to copy the contents of the wrong directory unless full and explicit path names are specified. Of course, being in the wrong directory is a pretty elementary mistake, but I'd guess that most hard disc users, if not a few floppy users, would admit they've done it at least once.

This prompts me to mention another easily overlooked command, 'CD'. As you know this means 'change directory' if a directory name is supplied with it, but how many of you remember that it also means 'current directory' and displays the current path when it's issued on its own? It can be very useful when you're swapping frequently between different discs and directories (and the phone rings so that you forget where you'd got to when you get back, as usually happens to me). You can also apply it to other drives, so that:

```
cd b:
```

will display the current path for drive B:, so it's always easy to check where you are. The reason that this facility is under-used is, I think, that the BBC micro has no such facility. The current directory isn't visible, short of cataloguing the discs.

I used copy to illustrate these short-cut path names, but they can also be used with other disc commands where appropriate. For example, to delete files in the current directory in drive B: you can type:

```
DEL B:
```

because again the '*.*' isn't necessary. Equally you can use double dots, so that:

```
DEL B:..
```

deletes all files in the parent of the current directory on drive B:, while:

```
CD B:..
```

will change the current directory in drive B: to the parent of the previous one. Obviously you need to take care with some commands, but used sensibly they can save a lot of typing.

That's it for this month, we'll continue with more DOS command facilities in the next Forum.

Wordwise Plus Auto-Backup Utility

Brian Herbert describes a Wordwise Plus segment program which will automatically maintain the last three generations of any text file as security against failure or corruption.

The object of the Wordwise Plus segment program listed here is to allow Wordwise text being edited in the main working area to be stored to disc at the press of a key, but retaining the previous version as a back-up. In fact, three generations of given text file will be kept, traditionally referred to as *son*, *father*, and *grandfather*. They are here shortened to 'S.' 'F.' 'G.' so that each generation may be stored in a different directory when using the DFS or ADFS (for the latter, the directories must be created in advance).

To enter the program, select Wordwise, and type the program into segment 0. Then save this with the name *AutoSave* (or similar). To use the program at any time, it should be reloaded into segment 0. Pressing Shift-f0 will then allow text to be saved at any time and with a minimum of effort, whilst still keeping earlier versions of the text for retrieval if required. This is most useful should more recent versions be corrupted, or parts be inadvertently deleted. Indeed, it is sometimes easier to return to an earlier version than undertake significant re-editing.

The program's logic for storing a new generation of a file, called TEXT say, is as follows:

1. If G.TEXT exists, it is deleted
2. If F.TEXT exists, it is renamed G.TEXT
3. If S.TEXT exists, it is renamed F.TEXT
4. The latest version is saved as S.TEXT

The *Autosave* program may be invoked either from the Wordwise menu, or from the text screen by typing Shift-f0, and following the screen instructions. You can abort the process, or choose a different file name from the current one. The program will also check that the text has been previously saved, and ask for a new file name if one is needed.

WARNING

The three generations of stored text should never be locked using *ACCESS. Should this be done, the *RENAME part of the program will fail. Care must also be taken neither to exceed the quota of files for the filing system, nor to fill the disc, so causing the "Disc full" or "Cat full" error messages to be given.

CONVERSION FOR ADFS OR OTHER DRIVES

The segment program as listed will automatically assume that drive 0 (DFS) is to be used. If this is not the case, change the first line of PROCDISC accordingly. For example, to use drive 0 (ADFS) replace **DRIVE 0* with **DIR :0* (and change the fifth to last line if appropriate). This assumes the appropriate disc filing system has been selected beforehand.

TECHNICAL DESCRIPTION

The operation of the program relies on the fact that the working file name (with directory) is automatically stored in F$, and may be used within the Wordwise language. It is, however, necessary to manipulate F$ to extract the file name without the directory letter. Unfortunately, the Wordwise Plus language does not allow manipulations such as MID$, LEFT$ and RIGHT$ (although Wordwise Plus 2 does) and other means must be used, as outlined below. For example:

```
SELECT SEGMENT 9
CURSOR BOTTOM
TYPE F$
```

will write the contents of F$ at the bottom of segment 9, finishing with the cursor on the space after F$.

To find out if the second character of F$ is a full stop, the following is added to the above program:

```
L%=LEN(F$)
CURSOR LEFT L%-1
IF GCT$="." THEN PRINT "SECOND CHARACTER
IS A FULL STOP"
```

The actual file name may now be picked out by reading each character as the cursor is moved to the right.

```
REM Wordwise Plus Program AutoSave
REM Version 1.0
REM Author  Brian K. Herbert
REM BEEBUG  June 1990
REM Program subject to Copyright

PROCEMPTY
SELECT TEXT
DISPLAY
END

.EMPTY
IF H$<>"" THEN F$=H$
A%=1
IF F$="" THEN A%=0
IF A%=1 THEN PROCFIND
IF A%=0 THEN PROCNEW1
PROCMENU
ENDPROC

.NEW1
CLS
VDU31,8,7
P. "File name undefined"
VDU31,11,9
P. "Press any key"
G%=GET
Z$=""
PROCNEW
ENDPROC

.FIND
L%=LEN(F$)
SELECT SEGMENT 9
CURSOR BOTTOM
TYPE F$
CURSOR LEFT L%-1
B%=0
IF GCT$="." THEN  B%=1
IF B%=1 THEN C%=L%-2
IF B%=0 THEN C%=L%
IF B%=0 THEN CURSOR LEFT 2
Z$=""
I%=0
REPEAT
I%=I%+1
Z$=Z$+GCT$
UNTIL I%=C%
DELETE LEFT L%
B$=Z$
ENDPROC

.MENU
REPEAT
Y$="S."+B$
X$="F."+B$
W$="G."+B$
REPEAT
CLS
VDU31,4,6
P. "Save file with name = ";
P. B$;
P." ?"
VDU31,15,9
P. "Yes"
VDU31,15,11
P. "No"
VDU31,15,13
P. "Abort"
D%=ASC(GCK$)
IF D%>90 THEN D%=(D%-32)
UNTIL D%=89 OR D%=78 OR D%=65
IF D%=89 THEN PROCDISC
IF D%=78 THEN PROCNEW
IF D%=65 THEN PROCABORT
UNTIL D%=89 OR D%=65
ENDPROC

.ABORT
CLS
VDU31,11,7
P. "File not saved"
VDU31,11,9
P. "Press any key"
G%=GET
ENDPROC

.NEW
REPEAT
CLS
VDU31,2,7
P. "File was loaded as name = ";
P. Z$
VDU31,2,9
P. "New file name = S.";
A$=GLK$
UNTIL LEN(A$)>0 AND LEN(A$)<8
B$=A$
ENDPROC

.DISC
*DRIVE 0
E%=OPENIN(Y$)
```

Continued on page 52

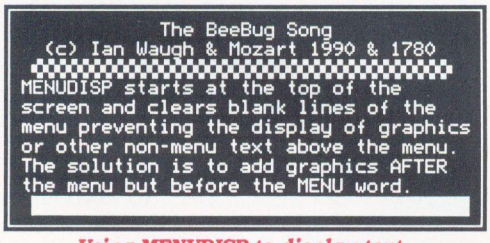
Music Programming In Ample (Part 3)

Ian Waugh concludes this short series on programming in Ample with some ideas on presenting your compositions in an attractive style.

I hope you found the ideas and techniques discussed in the last two articles informative and useful in your music. We'll end this short series by looking at ways of presenting your music and how to move through the Ample environment from within a program.

CHAINING PROGRAMS

We'll begin by tackling the thorny problem of chaining pieces. As you will have discovered, if you load a program while one in memory is already playing, the new program will completely replace the old one so you can't use 'overlays' to produce one continuous composition (if you have succeeded in doing this, please let us know). But what you can do is to make one piece chain another so that a series of pieces will load from disc and play automatically, like an LP or cassette. As with most things in Ample, there is more than one way to do this.



Using MENUDISP to display text

You can't simply insert an instruction to load and run the next program at the end of one of the music parts because control will pass through to the instruction before the piece has finished playing. We have to hold up execution of the instruction until the music has finished. We can do this with the help of two Ample words - QTIME and IDLE. In order to execute music instructions, Ample reads a little way ahead. QTIME returns the difference between the player's time and the system's time. During play this will be positive. IDLE does as its name suggests - nothing. It is used simply to mark time. For example, here's a word which waits for you to press the Ctrl key:

```
"Cwait"[REP( -2 QKEY )UNTIL( IDLE )REP]
```

Alter the QKEY value for other keys.

NOTE: All the programs listed with this article are written in Ample and may be used only with the Music 5000 system produced by Hybrid Technology.

Listing 1 - song

```
"RUN" []  "wait" []  "mix" []  "var" []
"part1" []  "part2" []  "chain" []

"RUN" ["12"PLAY
chain
]
"chain" [REP( var #? )UNTIL( IDLE )REP
$2 """song2""LOAD RUN"
]
"mix" [M5MIX
1 SHARE 1 VOICES
 1 VOICE Upright 128 VOL 0 PAN
2 SHARE 1 VOICES
 1 VOICE Upright 128 VOL 0 PAN
PNUM SHARE
]
"part1" [OFF var #!
SCORE 24,
0: CGeG cGeG dGfG cGeG
   cAfA cGeG bGdG cGeG ^
wait
]
"part2" [SCORE 24,
1: C/// E/G/ b//12,CD     24,c/ ^/
   A/// g/C/ g/8,fGf12,eF 24,e/ ^/
]
"var" [GVAR
]
"wait"  [REP( QTIME -100 #< )UNTIL( IDLE
)REP
ON var #!
]
```

Type in Listings 1 and 2 and save them as *song* and *song2* respectively. Don't run Listing 1 until you've saved it! With *song2* on disc, run *song*. It will play, and then *song2* will load and run. Let's

see how it works. *var* is defined as a variable, and simply acts as a flag. It is turned off by *part1* as soon as the program is run. *wait* sits IDLEing until 100 ticks past the last event (to give any sustained notes a chance to die away), and then it turns the *var* flag on. During this time, the RUN word instructs the music parts to play and would normally return to the % prompt, but it has been caught by the chain word. This is IDLEing, too, but it's waiting for *var* to be switched on. When it is, control passes through the REP(...)UNTIL(...)REP loop to the following line which loads and runs *song2*. $2 is used to discard the existing command line. If you leave this out, a mistake will be reported after *song2* has loaded.

Listing 2 - song2

```
"RUN" []  "mix" []  "part1" []
"part2" []

"RUN" ["12"PLAY
]
"mix" [M5MIX
1 SHARE 2 VOICES
 1 VOICE Upright 128 VOL 0 PAN
 2 VOICE Upright 128 VOL 0 PAN
2 SHARE 1 VOICES
 1 VOICE Upright 128 VOL 0 PAN
PNUM SHARE
]
"part1" [SCORE 48,
0: F^^f(C) e(C)^(^)^(^)e(C)
   d(C)^(^)^(^)d(b) c(C)^(^)c(C)^(^)
]

"part2" [SCORE 12,
0: A/BCDEFGAgfedcba g/ABCDEFGfedcbag
   f/GABCDEFedcbagf e/FGABCDc///^///
]
```

This method of chaining programs is suggested in the *Ample Nucleus Programmer Guide* and it works fine. But you'll notice that while the program is waiting, you don't have any control over the keyboard, and you can't use the editors. Let's look at an alternative method. Remove *wait* and *var* from the song program entirely. Redefine *chain*, and define a new word:

```
   "chain" [REP( QTIME -100 #< )UNTIL(IDLE
)REP
   $2 """song2""LOAD RUN" tab ]
   "tab"[&900 138 -12 CODE #2 #2 ]
```

Put *tab* on the first line of *part1* and *chain* on the last line (we'll look at the *tab* word in a moment). Save it as *song1*. You'll probably be in the Notepad, so type RUN. You'll see the cursor jump into the edit area, the music will play, the cursor will jump out to the command line and *song2* will load and play. However, if you are not in an editor when *chain* is activated, it won't work. You can use either method of chaining depending on what you require. The first is useful if you don't want any listener interaction/interference!

SYNCHRONISING TEXT AND MUSIC

Many Hybrid Music System owners use their system to create arrangements of pop songs and it seems like a good idea to have the lyrics appear on screen as the music plays. If the lyrics are inserted in the 'right place', beside the notes, they will simply race ahead of the music. This is another problem we can solve using QTIME.

Listing 3

```
"o" [$OUT
]
"w" [REP( QTIME 0#< )UNTIL( IDLE )REP
]
"part9" [1, ^w 24,
"This "o ^^^^w "is "o ^^w "the "o ^^w
"Beeb"o ^^^w "Bug "o ^w "Song."o ^^^^w
NL
"See "o ^^^^w "how "o ^^w "it "o ^^w
"scro"o ^^w "lls "o ^w "a"o ^w "long."o
]
"RUN" ["129"PLAY
]
```

Load *song* (or *song1*) and REM out or remove the words which chain *song2*. Add to this the words in Listing 3 (don't forget the modified RUN word) and run it (apologies for the lyric content but it serves to illustrate the procedure!). The character 'o' is simply used as a shorthand method of printing the text. The 'w' is an IDLE word - literally. The rests mark time between the notes and have been arranged to match up with the note output of *part1*. You will notice a short rest - 1, - at the start of *part9*. This is to hold up the output of the first word, "This". It will actually appear one tick later than it should but

it isn't noticeable in this example. If timing is crucial you can insert an equivalent rest at the start of the other music parts.

This method can be used to synchronise anything to the music, including other sounds or animation. This is not as difficult as it may appear. Characters can be defined using VDU23 (23 #OUT in Ample) and printed to the screen with #OUT. For a very simple demonstration, redefine 'o' as follows:

```
"o" [23#OUT 224#OUT 8#OUT 12#OUT 10#OUT
9#OUT 8#OUT 120#OUT 120#OUT 112#OUT
31 #OUT 34 RANDL #OUT 24 RANDL #OUT
224 #OUT SP
$OUT
]
```

Insert '4 MODE' at the start of *part9*. It's hardly Fantasia, but then it only took a few minutes to write. The first two lines define character 244 as a note, the next line uses VDU31 to site the cursor at a random position on the screen, the fourth line prints the note followed by a space, and the last line prints the lyrics. Perhaps you could write a simple game in Ample - and think of the sound effects!

OSCLI

Ample supports OSCLI commands, and they are very easy to implement. You can type in * commands in command mode at the % prompt, but within a word you must use OSCLI. If you want to catalogue a disc from within a program, here's the word:

```
"cat"["CAT" OSCLI ]
```

You can also use OSCLI to load a screen created with an art or graphics program which could be an impressive loading screen for a music album. The following will load a screen into mode 3 (you must enter mode 3 first, of course):

```
"pic" ["LOAD file 4000" OSCLI ]
```

ENVIRONMENTAL CONTROL AND MACROS

Let's move on now and see how we can move through the Ample environment from within a program. Listing 4 demonstrates the process so type it in and run it. You will see the program go to the main menu, select Notepad, GET Upright into it and step along to the amplitude waveform.

Listing 4

```
"RUN" []  "key" []  "$key" []

"RUN" ["MAIN" $key 13 key
142 key % Cursor down
13 key  % Return
"""Upright""GET"$key 13 key
9 key   % Tab
141 key 141 key 142 key
]
"key" [#B12 138 &FFF4 CODE #2 #2
]
"$key" [LEN FOR( 1$- ASC key )FOR $2
]
```

The *tab* word used earlier uses CODE to call a machine code subroutine which pokes the value for the Tab key into the keyboard buffer. The word *key* in Listing 4 is a general purpose word for poking values into the keyboard buffer. *$key* is used for words (text) which are converted into ASCII values letter by letter before *key* is called to poke them, too, into the buffer. Note the value of 142 which is used for the down cursor key. You might have expected this to be 138 but 142 is the value assigned by Ample.

You can use this technique to create macros for a variety of applications - to enter an editor before playing a piece of music, to GET a word into an editor or to move around within an editor. You could use it if you find you are constantly re-editing a word during program development or to force the program into the Mixing Desk, for example, if you want the listener to run it from there.

One key which can't be simulated from within a program is the Shift key. So having moved to the waveform in the above example, you can't make the program 'press Shift' and select another waveform. If you've discovered how to do this, please let me know.

MENUS A LA CARTE

When you run a program it's traditional for a display screen to appear to tell the listener something about the music and who wrote it.

The screen may also hold a list of choices - a menu - from which a selection can be made. The jukebox menus which are loaded from Hybrid discs by pressing f9 immediately spring to mind.

Listing 5

```
"cls" [26 #OUT   % Restores default
windows 12 #OUT          % Clears text
area
]
"menu1" [7 MODE MENUDISP
%
%
%
%MENUDISP starts at the top of the
%screen and clears blank lines of the
%menu preventing the display of graphics
%or other non-menu text above the menu.
%The solution is to add graphics AFTER
%the menu but before the MENU word.
%        RUN the program %"129"PLAY
30 #OUT DISPLAY
%            The BeeBug Song
%   (c) Ian Waugh & Mozart 1990 & 1780
150#OUT"ffffffffffffffffffffffffff"$OUT
"ffffffff$"$OUT  MENU
]
"menu2"  [clsNL23FOR(134#OUT  157#OUT
132#OUTNL)FOR
%Put Blue New Backgrnd codes down left
28#OUT  3#OUT  23#OUT  38#OUT  2#OUT  30#OUT
%VDU 28 - Define Text Window
NL NL NL DISPLAY
%          The BeeBug Song
%                by
%        Ian Waugh & Mozart
%
%info available
"129"PLAY
]
"menu3" ["129"PLAY
cls NL NL NL 131 #OUT DISPLAY
%          The BeeBug Song
%
%
%                        info available
]
"press" [NL 132 #OUT 157 #OUT 135 #OUT
                % New Blue background
10 FOR( SP )FOR % Spaces to center text
"Press RETURN" $OUT
$IN  $2          % Wait for keypress
%menu
]
```

The construction of menus is described quite well in the Music 5000 User Guide so I won't go over the basics again. The three main words used are DISPLAY, MENUDISP and MENU. Listing 5 contains some ideas for varying your menu displays. The first word, *cls*, restores default windows and clears the text area. Ostensibly this is to clear the screen ready for a menu display, but it is also useful if you want to TYPE a word and see it in full when in an editor. You could insert a mode change instruction here, too.

menu1 is self-explanatory and shows how to add graphics above the menu options. MENUDISP would normally remove these. *menu2* may look more complicated but all it does is to 'stripe' the left of the screen with teletext codes so that the display is more colourful. For something simpler, *menu3* just colours the heading.

The final word is very useful if you have more than one screen of information to display. It can be used to call a word specifically - in which case you would unREM the menu word on the last line (or substitute whichever word you wished). But by not specifying a word to go to, it can be used by several words, in which case the body of a menu word may look like this:

```
%
%   Menu option 1 % opt1 press
%   Menu option 2 % opt2 press
%   Menu option 3 % opt3 press menu
%   Menu option 4 % opt4 press menu
%
```

Options 3 and 4 would go back to the menu.

LP AND CD MENUS

Many Ample albums use the jukebox function (f9) to display a menu of the pieces. It would be easy to create an 'LP' menu in which you could 'put the needle' on any of the pieces and have them play through to the end - or even cycle continually from the top. Perhaps a more interesting idea would be to create a 'CD' menu which lets the listener select the order in which the pieces play. Hint: you'd need to use an area of memory which is not cleared when a new program is loaded - such as &8E and &8F in page zero.

Finally, on the subject of menus, being of an inquisitive nature I really like to know a little about the Ample programs I listen to and the people who write them - why the program was written, what problems had to be overcome during programming, any special techniques used and so on. Traditionally, such information is held in an *info* word, and one of my pet hates is to type *info* and be greeted with 'Mistake'! So this is a plea to all programmers to include information in your programs.

IN CONCLUSION

I hope this short series has given you an insight into some of the more advanced things you can do when you move beyond the editors and start to explore the Ample language. I've made several references to the Ample Nucleus Programmer Guide (£19.95 from Hybrid) and this is essential reading for the Ample programmer.

A good way to investigate programming techniques is to examine other people's programs. Other than the Ample albums released by Hybrid, Panda and others (see reviews in BEEBUG Vol.8 No.10), an excellent source of new ideas is available by joining the AMPLINEX user group which produces a bi-monthly magazine on disc - and written in Ample. It costs a nominal £5.00 to join and each disc is £2.00 UNLESS you contribute to that issue (anything from an article to an instrument definition) in which case it is free! I can't recommend AMPLINEX too highly.

Finally, if there are any aspects of programming in Ample you would like us to cover in a future series or if you want to comment on this series, please write to the editorial address. If there is sufficient interest then it may be possible to to provide further coverage of the Ample scene.

ADDRESSES

Hybrid Technology Ltd.,
273 The Science Park,
Cambridge CB4 4WE.
Tel. (0223) 420360.

AMPLINEX,
26 Arbor Lane, Winnersh,
Berks RG11 5JD.

B

Wordwise Plus Auto-Backup Utility (continued from page 47)

```
CLOSE# E%
IF E%>0 THEN PROCSAVE
IF E%=0 THEN PROCDISC1
ENDPROC

.DISC1
REPEAT
CLS
VDU31,5,6
P. B$;
P. " = file name is not in"
VDU31,5,8
P. "directory S. on this disc."
VDU31,12,11
P. "Carry on and save"
VDU31,12,13
P. "Return to menu"
F%=ASC(GCK$)
IF F%>90 THEN F%=(F%-32)
UNTIL F%=67 OR F%=82
IF F%=67 THEN PROCSAVE
IF F%=82 THEN D%=0
ENDPROC

.SAVE
```

```
CLS
VDU31,10,11
P. "Now saving file"
M%=OPENIN(W$)
N%=OPENIN(X$)
P%=OPENIN(Y$)
CLOSE#0
IF M%>0 THEN OSCLI "*DELETE "+W$
IF N%>0 THEN OSCLI "*RENAME "+X$+" "+W$
IF P%>0 THEN OSCLI "*RENAME "+Y$+" "+X$
SELECT TEXT
SAVE TEXT Y$
CLS
VDU31,3,7
P. "FILE SAVED WITH NAME = ";
P. B$
VDU31,10,9
P. "IN DIRECTORY = S"
VDU31,10,11
P. "IN DRIVE = ZERO"
VDU31,10,13
P. "Press any key"
G%=GET
ENDPROC
```

B

Adventure Games

by Mitch

Product	**LAST DAYS OF DOOM**
Supplier	**Topologika**
	PO Box 39, Stilton,
	Peterborough PE7 3RL.
	Tel. (0733) 244682
Price	**£19.95 + 50p p&p (all formats)**

I'm beginning to get seriously worried about your mental condition. Only a fool would have returned to Doom after your last fiasco. Yet here you are again, bright-eyed and bushy-tailed, just begging for something awful to happen. Believe me, you won't have long to wait!

The planet Doom is dying - any fool can see that. The crust is shuddering as if it were alive, and belching volcanoes are turning the purple sky black. Any adventurer with half a brain would have turned the ship around, pointed its nose at the nearest star cluster, and hit the hyperdrive. But not you! Just because you survived the first two episodes of the Doom Trilogy, (*Countdown* and *Return to Doom*) it doesn't mean that you are invincible. It's also fair to observe that crash-landing your ship into a crevasse is hardly an auspicious beginning to this final quest.

There is of course an 'up-side' to all of this gloom and 'Doom'; you do after all have your trusty robot dog to help you. Although he tends to be as much good as a chocolate tea-pot most of the time, he does understand 'Fetch' so I suppose he will come in handy.

Once you appreciate that there are armed robots, all manner of fanged space creatures, and lethal electronic defence mechanisms all impatiently awaiting your arrival, you will begin to get the message that the 'Welcome Mat' is not an item which has gained much popularity on this planet. In this all-text adventure the writers have adopted the motto - 'Never give a sucker an even break', and the puzzles range from hard to mind-bending.

Not being content with the usual 'Get' and 'Use' style of puzzles, the authors seem to be pushing our concepts of adventures into new realms. Some of the mental challenges appear to be derived as a result of conversations with philosophers who have spent their declining years discussing the number of angels who should theoretically be able to balance on the point of a pin. The upshot is a game which I suspect will tax the average player considerably, which is fine if you appreciate what you are getting into, but otherwise it just might break your heart.

Much of the action takes place amidst the volcanic landscape of the planet, with excursions into the ruined buildings of the long dead inhabitants. Players of the previous Doom games will recognise the familiar characters and objects. The robots are back, as is the black metal rod with the rusty star on the end. With typical Topologika humour, the rod which had one use in the first Doom game and two uses in the second, now turns up with three uses. The game has a surprising conclusion which will certainly catch you unprepared, but it's going to take a long time to see it.

Topologika has a record for quality adventures, and this game follows the tradition. The standard Help system is included on the disc, and you can use this to obtain increasingly specific nudges when you finally have to admit temporary defeat.

Well, I feel that I have given you all the advice that should be given. However, I can see from the gleam of insanity in your eye that you are impervious to reason, so I wash my hands of you. I wish you luck because you are going to need it.

Bulletin Boards

Following on from last month's list, here is a further selection of bulletin boards which should contain material of interest to BBC users. Do please let us know if you run a board, or know of one, which has not been listed.

Please remember that nearly all these boards are run in the sysop's spare time for no profit.

Board:	**Advance Opus**
Location:	Hull
Tel. number:	(0482) 586285
Speeds:	V23
Access times:	24 hours

Board:	**Co-op Board**
Location:	London
Tel. number:	081-316 6488
Speeds:	V22/22b/23
Access times:	24 hours

Board:	**The Desert**
Location:	Chester
Tel. number:	(0244) 550332
Speeds:	V21/22/22b
Access times:	24 hours

Board:	**Gaslamp**
Location:	Rochdale
Tel. number:	(0706) 358331
Speeds:	V21/22/22b/23
Access times:	24 hours

Board:	**Gnome at Home**
Location:	London
Tel. number:	081-888 8894
Speeds:	V23 viewdata
Access times:	24 hours

Board:	**I.A.D.**
Location:	Merseyside
Tel. number:	051-733 1135
Speeds:	
Access times:	19.00-23.00

Board:	**Kirklees Opus**
Location:	Huddersfield
Tel. number:	(0484) 665415
Speeds:	V21/22/22b/23
Access times:	24 hours

Board:	**Lambda**
Location:	Edinburgh
Tel. number:	031-556 6316
Speeds:	V21/22/22b/23/23v
Access times:	24 hours

Board:	**Mininet**
Location:	Cleveland
Tel. number:	(0642) 672813
Speeds:	V21/22/22b/23
Access times:	24 hours

Board:	**North Yorks QBBS**
Location:	Nidderdale
Tel. number:	(0423) 868065
Speeds:	V21/22/22b/23
Access times:	24 hours

Board:	**Peacenet**
Location:	Uxbridge
Tel. number:	(0895) 448998
Speeds:	V23 viewdata
Access times:	24 hours

Board:	**Phantom**
Location:	Barnsley
Tel. number:	(0226) 340425
Speeds:	V21v/22v/23v
Access times:	24 hours

Board:	**Rivendell**
Location:	Nottingham
Tel. number:	(0602) 640488
Speeds:	
Access times:	22.30-06.00

Board:	**The Sin Bin**
Location:	Leeds
Tel. number:	(0532) 661536
Speeds:	V23 viewdata
Access times:	24 hours

Board:	**Sirius**
Location:	London
Tel. number:	081-542 3772
Speeds:	V21/22/22b/23
Access times:	24 hours

Board:	**Tug II**
Location:	Droitwich
Tel. number:	(0905) 775191
Speeds:	V21/22/22b
Access times:	22 hours

Make a Date in Your Diary For the

*New Products
*Old Favourites
*Workshops
*Seminars
*Demonstrations
*Daily Competitions
*Free Draws
*Fabulous Prizes

Find us close to Victoria Station at
The Westminster Exhibition Centre
(Horticultural Hall)
Elverton Street, London SW1

Friday 7th Sept .... 12 noon to 7.00pm
Saturday 8th ..... 10am to 6.00pm
Sunday 9th ..... 10am to 6.00pm

In association with
BBCACORN
USER
MAGAZINE

Complete Range of BBC ACORN Hardware and Software

BOOK NOW, SAVE MONEY

For advance tickets fill in the coupon below and send with remittance to:-
SAFESELL EXHIBITIONS
Market House, Cross Road, Tadworth, Surrey KT20 5SR

Advance tickets:- Adults £3.50 Under 16's £2.50 At the door:- Adults £4.00 Under 16's £3.00
Please send ..... Adult tickets at £3.50 each. Please send ..... Under 16's tickets at £2.50 each
I enclose cheque / P.O. for £..... Payable to **Safesell Exhibitions Ltd.**

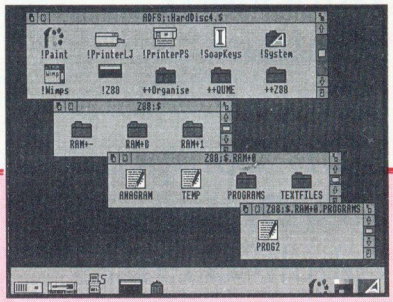
DON'T SEND CASH !!

PLEASE ENCLOSE A STAMPED ADDRESSED ENVELOPE

RISC USER

The Archimedes Magazine & Support Group

Risc User is enjoying the largest circulation of any magazine devoted solely to the Archimedes range of computers. Now in its third year of publication, it provides support to schools, colleges, universities, industry, government establishments and private individuals. Existing Beebug members, interested in the new range of Acorn micros, may either transfer their membership to the new magazine or extend their subscription to include both magazines.
A joint subscription will enable you to keep completely up-to-date with all innovations and the latest information from Acorn and other suppliers on the complete range of BBC micros. RISC User has a massive amount to offer to enthusiasts and professionals at all levels.



Here are just some of the topics covered in the most recent issues of RISC User:

INTRODUCING C
A wide ranging new series on C, a major programming language for the Archimedes.

WATCHDOG: ANTI-VIRUS
A Desktop application to help protect your hard and floppy discs against viruses.

DESKTOP HOTKEYS: KEYBOARD WINDOW CONTROL
A Desktop application which allows you to manipulate windows on the screen directly from the keyboard, using hotkey sequences.

GENESIS
A review of this impressive software package for generating multi-media information systems.

ARC TO Z88
The second article on using the Arc in conjunction with small portable computers.

ASSEMBLER WORKSHOP
A major series for the more advanced ARM processor programmer. The latest one offers a program which displays information on parameters passed by a CALL statement.

SCANNERS FOR THE ARC
A review of three different scanners.

MASTERING THE WIMP
A major series for beginners to the Wimp programming environment. The latest issue is dedicated to Icons.

CD ROM FOR THE ARCHIMEDES
A preview of the latest innovation for the Archimedes.

INTO THE ARC
A regular series for beginners. The latest article is: Avoiding errors with typed listings.

ARCADE
A round-up of the latest games for the Arc.

As a member of BEEBUG you may extend your subscription to include RISC User for only £8.10 (overseas see below).

Don't delay!
Phone your instructions now on (0727) 40303

Or, send your cheque/postal order to the address below. Please quote your **name and membership number.** When ordering by Access, Visa or Connect, please quote your card number and the expiry date.

SUBSCRIPTION DETAILS

Destination	Additional Cost
UK,BFPO &Ch Is	£ 8.10
Rest of Europe and Eire	£12.00
Middle East	£14.00
Americas and Africa	£15.00
Elsewhere	£17.00

RISC User, 117 Hatfield Road, St Albans, Herts AL1 4JS, Telephone (0727) 40303, FAX (0727) 60263

Z88 TO BBC MICRO

Colin Pither

When sending data from the Master or BBC micro to the Z88 using the BEEBUG Z88 - BBC Link software, the user is able to direct the output to whichever RAM slot is required. However, when receiving data, the choice of drive on which to store the data is not provided. Unless you select the option to change the incoming filename, you have to specify the drive each time a new file is transmitted. The insertion of the following lines into the *Talk* program on the Link disc will provide the choice of drive whichever way files are received:

```
  392 PRINTTAB(0,6)"Which drive to receive
data? ";
  393 d$=GET$:IF d$<CHR$(48) OR d$>CHR$(
49) THEN 392
  394 OSCLI("MOUNT "+d$)
```

To ensure that control is returned to drive 0 on completion of the file transfer, insert the further line:

```
  127 *MOUNT 0
```

The request for a drive number will accept only 0 or 1, which is correct for a Master using the ADFS. If you are using the DFS then increase the CHR$(49) in line 393 to CHR$(51) and replace 'MOUNT' in lines 394 and 127 with 'DRIVE'. Because the new statement uses screen line 6, the original line 395 will need the parameters of the TAB instruction increased to (0,7).

The Link program can be further enhanced by programming otherwise unused function keys at the commencement of the *Talk* program to echo repeatedly used directories, file names etc. A keystrip can then be made to reflect the action of each function key used.

BEEBUG TOOLKIT ON THE MASTER

Charles Seager

There is one way of using the BEEBUG Basic Toolkit on the Master. If you have a Basic II ROM installed, 'unplug' Basic IV, and 'insert' Basic II, and Toolkit will then work.

RECONFIGURING THE MASTER

Andrew Rowland

When it's time to change the Master's battery, it is sometimes suggested that you spool the configuration settings to disc. However, they cannot be reset by EXECing the spool file! Instead, use listing 1 to read bytes 1 to 50 of the CMOS Ram, where the configuration data is stored, and save them to a file called CMOS.

Then enter and run listing 2. This creates a utility called RESET, which does the reverse of listing 1, except in machine code (you cannot be sure that Basic will be available).

Finally create a boot file which *RUNs RESET. If your configuration settings become lost, corrupted or tampered with, simply boot the disc using Shift/D/Break. On a network, the station number must also be set up with the appropriate utility.

Listing 1

```
  10 REM .>SvCMOS
 100 FOR I%=0 TO 49
 110 B%=FNA(I%)
 120 I%?&1900=B%
 130 NEXT
 140 *SAVE CMOS 1900+31
 150 END
 160 :
1000 DEFFNA(X%):A%=161
1010 =(USR(&FFF4) AND &FF0000) DIV &10000
```

Listing 2

```
  10 REM .>RESETba
  20 :
 100 oswrch=&FFEE:osbyte=&FFF4
 110 oscli=&FFF7:mc=&900
 120 FOR pass=0 TO 3 STEP 3
 130 P%=mc:[OPT pass
 140 .cli
 150 EQUS "LOAD CMOS 1900":EQUB 13
 160 .entry
 170 LDX # cli MOD 256:LDY # cli DIV 256
 180 JSR oscli:LDX #0:STX &70
 190 .loop
 200 LDA &1900,X:TAY:JSR send
 210 INC &70:LDX &70:CPX #50:BNE loop
 220 LDA #7:JMP oswrch
 230 :
 240 .send
 250 LDA #162:JMP osbyte
 260 ]NEXT
 270 A$="SAVE RESET "+STR$~mc+" "+STR$~P
%+" "+STR$~entry
 280 PRINT A$:OSCLI A$
```

EDIKIT ROM

(incorporating Basic Booster)

An indispensible utility ROM for all Basic programmers for the Master, BBC B and B+ (all with sideways RAM). It contains eleven commands to help you with the editing and development of Basic programs:

***FTEXT** (find text) , ***FBASIC** (find Basic), ***FPROCFN** (find procedure/function) - three FIND commands, designed to help locate lines of programs according to their contents.

***LFROM** (list eight lines of a program), ***LPROC** (list procedure), ***LFN** (list function) - three commands, which list out significant segments of a program.

***RTEXT** (replace text) and ***RBASIC** (replace Basic) are find and replace commands directly analogous to FTEXT and FBASIC.

***SYSINF** (system information) - gives background information on the system variables and their sizes, together with the size of your program

***VARLIST** (list program variables) - lists variable names

***FKDEFS** (function key definitions) - prints out current function keys definitions.

```
>*Help Edikit

EdiKit 1.10

  FBASIC <element>
  FTEXT /<string>/
  FPROCFN
  LFROM <element>
  LPROC <procname>
  LFN <fnname>
  RBASIC <element>/<element>
  RTEXT /<string>/<string>/
  VARLST [%]|[<init letter>]
  SYSINF
  FKDEFS
  RENUMBER [<first>] [<last>] [<start>] [<inc>] [F]
  CHECKORDER [<first>] [<last>]
  BRMOVE [<first>] [<last>] [<destination>]
  BRCOPY [<first>] [<last>] [<destination>]
  SQUEEZE
  SMARTNUMBER [<proc base>] [<gap>]
  BLIST <filename>
  TEXTSAVE <filename>
  TEXTLOAD <filename>
>
```

```
3570LDA #ipbuff  S256:STA pptr
3580LDA #ipbuff e256:STA pptr+1:RTS
3590.reinitbptr
3600LDY#3:LDA(bptr),Y
3610CLC:ADCbptr:STAbptr
3610CLC:ADCbptr:STAbptr
3620LDA#0:ADC bptr+1:STA bptr+1:RTS
3620LDA#0:ADC bptr+1:STA bptr+1:RTS
3840LDY#1:LDA(bptr),Y:STA hi
3850INY:LDA(bptr),Y:STA lo
4090JSR inittptr:LDY#0
4110LDA(tptr),Y:BMI tokenfnd
4190CLC:ADC tptr:STA tptr
4190CLC:ADC tptr:STA tptr
4200LDA#0:ADC tptr+1:STA tptr+1
4200LDA#0:ADC tptr+1:STA tptr+1
4240.inittptr
4250LDA#ttable  S256:STA tptr
4260LDA#ttable e256:STA tptr+1:RTS
4340LDA(bptr),Y:CMP#nl:BEQ dtexit
4430.movetoknloop LDA(tptr),Y
4480INY:LDA(bptr),Y:ASL A:ASL A
```

Including the updated Basic Booster utilities:

SUPER SQUEEZE - A program compressor.

PARTIAL RENUMBER - A very useful utility which renumbers a selected block of lines.

PROGRAM LISTER - List any program direct from a file.

RESEQUENCER - Rearrange the lines in a Basic program - line numbering is automatically adjusted.

SMART RENUMBER - Renumber a program so that procedures start at a particular line number.

TEXTLOAD AND TEXTSAVE Save and load a Basic program as text.

EDIKIT (including Basic Booster) is available on:

		Members	Non-members	*Upgrade:	Members	Non-members
3.5" ADFS disc	Stock Code 1452a	£5.75	£15.00	Stock Code 1455a	£4.75	£6.33
40/80T DFS 5 .25"" disc	Stock Code 1450a	£5.75	£15.00	Stock Code 1453a	£4.75	£6.33
EPROM	Stock Code 1451a	£7.75	£18.00	Stock Code 1454a	£6.75	£9.00

*** If you have previously purchased Basic Booster, return your original disc or EPROM to obtain an upgrade at the reduced price.**

Please add 60p p&p (UK only). For overseas check the Membership page in this magazine.

Phone your order now on (0727) 40303

or send your cheque/postal order to the address below. Please quote your **name** and **membership number.** When ordering by Access, Visa or Connect, please quote your card number and the expiry date.

BEEBUG Ltd, 117 Hatfield Road, St Albans, Herts AL1 4JS. Telephone (0727) 40303.

POSTBAG

A SIMPLER 'CODE' KEY

The recent program which implements the equivalent of the Compact's Code key (see Vol.8 No.10) is a useful facility which also works with the new Master ROM. However, it seems, perhaps, over complicated by redefining the Shift/0 key. All that is required is the @ key, in that this key is rarely, if ever, used with the Shift key. The @ character is produced regardless of the setting of Shift or Caps Lock, and without the use of Shift.

I have therefore amended the original program to detect the input of Shift-@, and to convert this to code 250, subsequently to be used to add &80 to the ASCII code of the next key. The zero key is unaltered, and the program appears to work. Shift-@ followed by any other key produces the code of that key plus 128, and the character from the Master's extended character set.

To implement this, remove lines 220 to 300 from the original program and replace with:

```
220 LDA character:CMP #ASC"@":BNE exit
230 LDA #202:LDX #0:LDY #&FF
240 JSR osbyte:TXA
250 AND #8:BEQ exit
260 LDA #250:STA character
```

Neville Smith

RE-REGISTERING USED SOFTWARE

Buyers of used Computer Concepts Inter series of ROMs etc. may be glad to know that CC would appear happy for you to register as a second user of an original Computer Concepts software packages. Just write to them with the registration number and your name and address. I believe there is no charge for this.

Tim Sinclair

FINDING ROOM

I have recently received some discs of BEEBUG Vol.6 and am having trouble with two games, *Knight Quest* in No.10 and *Dracula* in No.6. In both cases I get the message 'No room' after a short while. I have a model B with Watford DDFS and shadow RAM board. Your advice would be appreciated.

J.D.Muddiman

It is not uncommon for programs to fail with 'No room' if memory is tight. A solution can often be found by setting PAGE to a lower value before loading the program. It is usually safe to set this to &1200 (simply type PAGE=&1200), provided no more than one disc file is open at any one time. If you have shadow RAM fitted, then ensuring that this is used should also resolve the problem. Locate any MODE statements in the program and add 128 to the mode number before resaving.

UPGRADING TO AN ARC

Having upgraded to an A3000 I was very interested in the articles on this subject in BEEBUG (see Vol.8 Nos.9 & 10). I have a View ROM and I find that this will not run satisfactorily when using 65Tube, but it will run with 65Host. However, I find this so slow that it is useless for serious work. 65Tube is a very satisfactory medium for suitable programs which appear to run much faster than on a BBC micro. I have both Inter-Word and Spellmaster both of which are fast and excellent pieces of software.

I would advise anybody who is thinking of upgrading, and has BBC programs which they want to continue to use, to see them working on an Archimedes first.

H.McDonald

The version of View supplied in ROM image format with the Master Compact works very well on the Archimedes using 65Tube. Unfortunately, this disc-based software is only available at a full price (£55.77 to members). CC have now introduced full ARM implementations of the Inter series, which are to be preferred to the originals. However, users contemplating an upgrade to an Archimedes should, perhaps, ask themselves whether it is worth it if they intend to continue using BBC micro software.

SAVING VIEW FILES WITHOUT RULERS

When attempting to use View to create the file W.DATES for use with the *Calenda* program (see BEEBUG Vol.7 No.7), it is quite easy to turn off formatting and justification, but the rulers are still present. Hence it seems to me that the program is not suitable for use with View, as claimed in the magazine.

C.A.Martin

The text can be saved from View, without including any rulers, by inserting markers at the start and end of the text to be saved, and then using the WRITE command instead of SAVE, thus:

```
WRITE W.DATES 1 2
```

Personal Ads

BEEBUG members may advertise unwanted computer hardware and software through personal ads (including 'wants') in BEEBUG. These are completely free of charge but please keep your ad as short as possible. Although we will try to include all ads received, we reserve the right to edit or reject any if necessary. Any ads which cannot be accommodated in one issue will be held over to the next, so please advise us if you do not wish us to do this. We will accept adverts for software, but prospective purchasers should ensure that they always receive original copies including documentation to avoid any abuse of this facility.

We also accept members' Business Ads at the rate of 30p per word (inclusive of VAT) and these will be featured separately. Please send us all ads (personal and business) to MEMBERS' ADS, BEEBUG, 117 Hatfield Road, St. Albans, Herts AL1 4JS. The normal copy date for receipt of all ads will be the 15th of each month.

WANTED: Acorn Master 512 upgrade complete with manuals, Gem discs, mouse etc, and of course reasonably priced, cash waiting. Tel. (0705) 371018.

Master 512 with Cumana CD800/S dual 40/80T drive and plinth, Viewstore, Interword and Spellmaster ROMs. Disc-based software such as Viewplot, Printerdriver, Office Master and Office Mate, games on cassette, User Guides and M128 reference manuals, bound BEEBUG magazines since vol. 3, all in excellent condition with original packaging. All for £650, Watford Beeb Video Digitiser plus ROM and software £60. Tel. (06285) 20320 eves and w/e.

Books; New Advanced User Guide £10, Master OS, A Dabhand Guide (disc & book) £10. Disc's; Lancelot, Time & Magik (both Level 9), PowerPlay £6 each, Play It Again Sam 4&6 £ 5 each, Exile, Elite, Jet Set Willy II £4 each, Galaforce, Icarus (master version) £3 each. Tapes; Star Seeker, Quest, Life of Repton, Acornsoft Hits II, Hibyte, Planetoid Conversion, Rik the Roadie, Rig Attack £2 each or all for £10. Home Entertainment Centre (Backgammon, Bridge, Chess etc) £10. ROMs; MOS+ for Master £7, Exmon II (latest Master compatible) £10, BEEBUG C £25, View Professional £40. Dumb terminal (mono screen, keyboard, RS232) £30, V23 modem with scrolling and viewdata software £25. **WANTED:** Music 5000. Tel. 01-698 3772.

BEEBUG magazines, complete sets from first issue to date with indexes, all in BEEBUG binders, mint condition, also Advanced User Guide in 'Cambridge' plastic binder. Bruce Smiths' Micro Assembly Language. Mini Office, Wordease & Masterfile II on disc, the lot only £30 (owner upgraded). Buyer collects from Preston Lancs. Tel. (0772) 717017.

Archimedes A310, RISC OS, manuals etc. boxed, as new - unwanted gift £545. Acorn colour monitor £195, will sell separately. Tel. (0234) 708463.

WANTED: 512 co-processor for M128. Tel. (0831) 126435.

M512/1024 (Solidisk PC+) with dual 40/80 D/S DD, Zenith mono monitor, View etc, cartridges; Master ROM, WW+, CP, Wordease, MFII, I-Word, I-Sheet, I-Base, DOS+ 2.1, GEM etc. Dabs M512 Guide, Disc & Shareware. See working. £500 o.n.o. Also BBC B with TORCH Z80 disc pack (dual D/S 40/80). BBC MOS, extra RAM and ROM box, Dataplus, Vufile, etc. CPM, Perfect Writer, Speller, Calc, Filer, BBC BASIC Z80 £350 o.n.o. Tel. 021- 308 0224.

Cumana CD200 dual disc drive, SS40T, mains powered, original packing, cable, manual, format disc, perfect. £80. Tel. (0327) 702095.

BBC B version 1.2 with shadow RAM, QFS, extension board, Cumana 40T 3.5" disc drive, £300 o.n.o. Tel. 071-736 5429.

Morley Teletext adaptor, 2 ROMs manual and disc £60. Tel. (0442) 826079 after 7pm (24hrs).

BBC B 32k issue 7 with Acorn DFS sideways ZIF & external speaker socket. Microvitec colour monitor, Watford 40/80T DS DD, Brother M-1009 printer (one print dot missing), Voltmace joystick, some software (Island Logic Music System, Watford Office Master, Signwriter), box of unused 3M discs and box of unbranded discs £300 o.n.o. will split. Tel. (0742) 314364 or (0602) 706502.

M128, Microvitec colour monitor, twin double sided 40/80T disc drive, ROM cartridge, ROMs, Mini Office II, Masterfax, Office Mate, AMX Super Art, Image Art Package, A4 Forms Designer, 6 Signwriter fonts, fontstyle, many games and utilities, over 150 discs, 3x disc boxes, manuals part 1&2, many books, bargain at £500. Tel. (0748) 833944.

BEEBUG Master ROM £15, Hershey Characters £7, Printwise II £10, Acorn Viewstore £20, Acorn Viewindex £5, Acorn PGD £5, Amx Extra Extra £10, FSE Admin Xtra £8, Chauffeur Utility £7. Tel. (0642) 311848.

Interbase, Intersheet, Interchart ROMs & manuals in perfect condition, £35, £20, £15. Tel. (0602) 722426.

Reference manuals parts 1&2 plus advanced reference manual, Morley AA board, 8 Master ROM cartridges, various makes. Reasonable offers. Tel. (04243) 4500.

WANTED: 8271 disc controller IC, Watford DDFS board with DDFS ROM, The original Acorn Music 500 or Hybrid's Music 5000. All offers considered. Tel. (0460) 74000 after 5pm or weekends.

Epson MX printer with manual, 'GRAFTRAX plus' together with 'Dumpout 3' printer dump ROM for BBC micro & books on graphics etc. £120. Tel. (0286) 870689 eves.

A3000 for sale still boxed, as new £549, E-Type, E-Type designer, Interdictor for sale, £14, £10, £24 respectively (o.n.o.). Tel. 081-560 7310.

WANTED: 512 board, pref. with software and accessories. Write to; A Urlberger, Waldheimstr. 57, 7302 Ostfildern-1, W. Germany.

BEEBUG Master Modem with command ROM. Offers? Tel. (0705) 464631.

WANTED: Accompanying 5.25" disc for the 'BEEBUGSOFT' members

booklet for reasonable price. Tel. (04882) 567 after 5pm or w/ends after 1pm.

New, cased, double-sided, Teac 40T disc drive, would swap for Interword and Intersheet ROMs. Tel. (0481) 56266 after 6pm.

M128 Turbo board £50, Intersheet, System Delta, Watford ROM Manager, all for M128. Also View 3.0 for B/B+ including printer-driver generator, boxed. All original ROMs with manuals. Offers? Tel. (0727) 67465.

BEEBUG magazines, complete from May 87 (Vol.6 Iss.1) to date (Vol.8 Iss.10). Offers? Tel. (0491) 873618.

For sale all with full original documentation. BBC B issue 7, Opus DFS, 16K Sideways RAM, Care ROM extension - £200 with free cassette player, Opus dual DS 40/80T DD including PSU £70, Z80 second processor, plus business software £90, Morley teletext adaptor £30, 64K Micron Plus EPROM Programmer £40, BEEBUG vols. 1-8 complete, plus magazine discs vols. 5-8, £100. Tel. (0264) 58659 weekends and after 6pm daily.

Morley Teletext adaptor, ATS & support ROMs, Manual £40, BEEBUG cassettes Vol. 5 £10, Vol. 6 £12, Vol. 7 £15, Vol. 8 £18. Tel. 01- 393 4630.

Archimedes A310 base + RISC OS, dual 3.5" disc drives, 5.25" disc drive interface, 2-slot backplane, manuals, box etc., PC Emulator, dual 5.25" disc drive, Akther, 40/80T + manual £150, buyer collects. Free to anyone who calls to collect; BEEBUG Vols. 3-7 inclusive, also Opus 3" disc drive with discs. Tel. (0276) 22031.

BBC Model B issue 7, £170. Tel. (0895) 621953 eves.

Archimedes A310 base with manuals, backplane and fan, and BEEBUG MkII external disc drive adaptor. Also some business software and a few games. Offers around £650. Tel. 081-540 8461.

Epson FX80, perfect working order, with handbook and model B cable £55, Genuine reason for sale. Tel. (0442) 56623.

M128 in Viglen case with double Opus 40/80T disc drives + Philips 7502 mono monitor + software £435 o.n.o. Also Taxan 625 colour monitor £295 o.n.o. all in excellent cond. Tel. (0268) 693770 eves.

BBC B issue 4, with 5.25" double DD reads 40/80 D/S density discs, Solidisk DFS and ADFS, Green monochrome monitor, Quest Paint ROM with mouse, Watford ROM/RAM board with 48K, ROMs include Dumpout 3 and Commstar. Joystick too, many disc games including Exile, Repton Infinity, Pipeline and Repton 1,2,3 A.T.W.I.F.S and Life of Repton. Tape recorder with leads, User Guide and Advanced User Guide along with many BEEBUG mags and Micro Users'. 5.25" lockable disc case with many utilities and discs, must sell all, new £1150, selling at£500. Tel. (0279) 814448.

Send an SAE for a list of **BBC computer books,** very reasonable prices. Titles from easy programming to machine code texts. Also some games tapes and discs, good condition, hobyist owner write to 118 Turners Road North, Luton, Beds.

Artroom, Ace, Digimouse £30, Chauffeur £5, Speech! £5, Genie Junior £10, bundle of children's programs (cassettes) £5, original version of Elite £3, Master ROM cartridge (ACP) £6, Books; Assembly Language Programming (Birnbaum) £6, Programming the 6502 (Zaks) £6, Creative Assembler (Griffiths) £3, Master sideways RAM User Guide (Smith) £6. Tel. (0302) 744005.

Archimedes floppy drive, boxed as new, only used for 5 months. Includes leads and new front panel. £95. Tel. (04024) 40838.

BEEBUG Magazines complete and in binders Vols. 1-3 £6 each, Vols. 4-7 £8 each. Worlds-Without-Words, educational pack 4-10 yrs, as new £20. Kinship (BBC Soft), family tree program for M128, as new £20. Tel. (0344) 55772 eves.

Convert your BBC Master into 512K and run most PC software using Acorn 80186 co-processor board complete with latest DOS plus, mouse and GEM Paint, GEM Draw & GEM Desktop. £125. Also Acorn Viewspell ROM £25 and ViewIndex disc £10, all complete with manuals and packaging. 086-732 8776.

BBC B ROMs, Commsoft, Beebdos, Wordwise, Okimate 20, Watford DFS, IFEL sideways RAM module each £10. First Word Plus, issue 1, unused £25. Tel. (0272) 730959.

WANTED: 512 co-processor, complete kit if possible, including software & case. Tel. (0703) 842646 (home), (0703) 662275 (work).

M128 computer £250, Microvitec 1451 colour monitor in plastic case £150, Turbo co-processor £50, Watford Electronics CLD800S 40/80 twin disc drive £100, BEEBUG Master ROM £10, Epson MX80FT3 printer £30. Tel. (04747) 7724.

M128 Keyboard case, connecting cable & dust cover £20, Beta Base & Utility (DFS) £12, Disc Doctor, damaged pin but operative £5, Sanyo cassette recorder D101 with cables £15, all original packing and manuals, lockable disc box with 25 ADFS reformatted discs, some rainbow £15, postage extra. Tel. (0932) 226076.

Computer Concepts RAM podule with 4X 32K RAMs £50 o.n.o. Watford Electronics Digitiser £100. (both for Archimedes) Tel. (0224) 535204.

Superart ROM, mouse & disc £35, Interword ROM £25, Delta 3b twin joysticks £10, 8 Superior Software disc games £50 (will split), 4ft BBC to KX-P1081 printer lead £3, BBC RGB (Digital) lead £1.50, 7 assorted cassettes £14 (will split), dust cover (Beeb) £2. Tel. (0727) 35877.

WANTED: Downloaded CEEFAX Telesoftware on disc, also BEEBUG mags. Vol. 1 numbers 1, 2, 3, 4, 5, & 8. Tel. (0482) 707099.

Master 512 with GEM software, Cumana 40/80T disc drive with PSU, AMX Super Art & Stop Press with mouse, PMS Publisher DTP package with 32 fonts, Acornsoft ISO Pascal cartridge & Viewspell ROM, BEEBUG Master & Dumpmaster ROMs, and Studio Eight disc, Interword ROM. ROMs housed in Care Electronics cartridges, a few disc based games, including Elite, plus over 40 blank discs in library box, less than 2 yrs old and in perfect condition, with manuals/boxes. Over £800 new, will accept £450. Tel. 081-952 8460 after 6pm.

Diablo Daisy Wheel Printer model 630 (Commodore badged) surplus to our requirements, serial interface, high quality 45cps, spare ribbons and daisy wheels, offers invited. Contact Mike Williams at BEEBUG, (0727) 40303.

BEEBUG MEMBERSHIP

Send applications for membership renewals, membership queries and orders for back issues to the address below. All membership fees, including overseas, should be in pounds sterling drawn (for cheques) on a UK bank. Members may also subscribe to RISC User at a special reduced rate.

BEEBUG SUBSCRIPTION RATES

		BEEBUG & RISC USER
£16.90	1 year (10 issues) UK, BFPO, Ch.I	£25.00
£24.00	Rest of Europe & Eire	£36.00
£29.00	Middle East	£43.00
£31.00	Americas & Africa	£46.00
£34.00	Elsewhere	£51.00

BACK ISSUE PRICES (per Issue)

Volume	Magazine	Tape	5"Disc	3.5"Disc
1	£0.40	£1.00	-	-
2	£0.50	£1.00	-	-
3	£0.70	£1.50	£3.50	-
4	£0.90	£2.00	£4.00	-
5	£1.20	£2.50	£4.50	£4.50
6	£1.30	£3.00	£4.75	£4.75
7	£1.30	£3.50	£4.75	£4.75

All overseas items are sent airmail. We will accept official UK orders for subscriptions and back issues, but please note that there will be a £1 handling charge for orders under £10 which require an invoice. Note that there is no VAT in magazines.

POST AND PACKING

Please add the cost of p&p when ordering individual items. See table opposite.

Destination	First Item	Second Item
UK, BFPO + Ch.I	60p	30p
Europe + Eire	£1	50p
Elsewhere	£2	£1

BEEBUG
117 Hatfield Road, St.Albans, Herts AL1 4JS
Tel. St.Albans (0727) 40303, FAX: (0727) 60263
Manned Mon-Fri 9am-5pm
(24hr Answerphone for Connect/Access/Visa orders and subscriptions)

BEEBUG MAGAZINE is produced by BEEBUG Ltd.

Editor: Mike Williams
Assistant Editor: Kristina Lucas
Technical Editor: Alan Wrigley
Technical Assistant: Glynn Clements
Production Assistant: Sheila Stoneman
Advertising: Sarah Shrive
Managing Editor: Sheridan Williams

CONTRIBUTING TO BEEBUG PROGRAMS AND ARTICLES

We are always seeking good quality articles and programs for publication in BEEBUG. All contributions used are paid for at up to £50 per page, but please give us warning of anything substantial that you intend to write. A leaflet 'Notes to Contributors' is available on receipt of an A5 (or larger) SAE.

Please submit your contributions on disc or cassette in machine readable form, using "View", "Wordwise" or other means, but please ensure an adequate written description is also included. If you use cassette, please include a backup copy at 300 baud.

In all communication, please quote your membership number.

ISSN - 0263 - 7561

Printed by **Newnorth Print Limited** (0234) 41111

Magazine Disc/Cassette

JUNE 1990 DISC/CASSETTE CONTENTS

SPLINE TEXT - A utility which allows you to print messages using outline or filled Helvetica characters, any scale or size up to 8"x7", on an Epson compatible printer.

TURMITE - This program creates an intriguing graphical display and delves into the world of mathematical turmites - close relations to mathematical worms.

STORING VARIABLES AND PROCEDURES IN SWR ON A MODEL B - Two utilities for storing and overlaying Basic procedures, functions and variables in sideways RAM.

FOG INDEX - An intriguing program, which allows you to evaluate the complexity and ease of understanding of any text.

A VERSATILE CHARACTER EDITOR - An application for defining new characters, graphics characters for icons and groups of characters for graphic designs.

PRACTICAL ASSEMBLER (Pt 2) - A program which demonstrates the use of macros in Assembly programming.

DECIMAL SQUEEZE - An entertaining and challenging program which tests your understanding of decimal numbers.

FIRST COURSE: Menu Routine - A program to demonstrate a menu system, which can be adapted for your own use.

DESIGNER SHOOT-'EM-UP (Pt 2) - A concluding program which allows you to create the screens for your game.

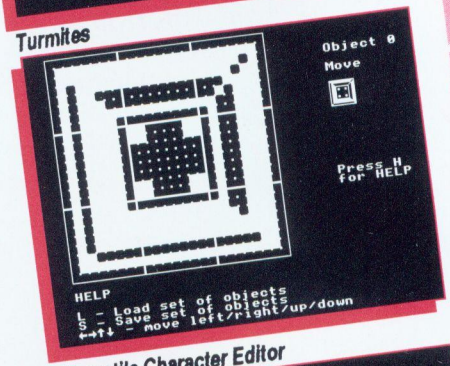
WORDWISE PLUS AUTO-BACKUP UTILITY - The last three versions of your text are automatically saved with this utility.

MUSIC PROGRAMMING IN AMPLE (Pt 3) - Five Ample programs, demonstrating chaining pieces, synchronising text and music, creating macros and menus.

MAGSCAN DATA - Bibliography for this issue (Vol.9 No.2).

*** SOLITAIRE** - A bonus item providing hours of entertainment.

Turmites



Versatile Character Editor

Designer Shoot-'Em-Up

ALL THIS FOR £3.50 (CASSETTE), £4.75 (5" & 3.5" DISC) + 60p P&P (30p FOR EACH ADDITIONAL ITEM)

Back issues (5.25" disc since Vol.3 No.1, 3.5" disc since Vol.5 No.1, tapes since Vol.1 No.10) available at the same prices.

SUBSCRIPTION RATES	UK ONLY		OVERSEAS	
	Disc (5" or 3.5")	Cassette	Disc (5" or 3.5")	Cassette
6 months (5 issues)	£25.50	£17.00	£30.00	£20.00
12 months (10 issues)	£50.00	£33.00	£56.00	£39.00

Prices are inclusive of VAT and postage as applicable. Sterling only please.

Cassette subscriptions can be commuted to a 5.25" or 3.5" disc subscription on receipt of £1.70 per issue of the subscription left to run. All subscriptions and individual orders to:

BEEBUG, 117 Hatfield Road, St.Albans, Herts AL1 4JS.

THE LEARNING CURVE

New from Acorn

THE LEARNING CURVE

NEW - FROM ACORN

It consists of:

An A3000 Computer
The DR-DOS PC Emulator
First Word Plus v2 Word Processor
Genesis Information System
A Video explaining how to use the package
Information on the National Curriculum

Save Over £185

The free software bundled with this package is worth £300, and yet the entire system costs only £115 more than the standard A3000.

Ideal For Home & Education

This system is tailor-made for the home user with children. The A3000 and other Archimedes systems are increasingly used in schools and this all-inclusive package with the excellent Genesis software will supplement Information Technology in the National Curriculum.

Just Right for Home & Business

A Word Processor is essential in all home systems, and 1st Word Plus with its built-in spelling checker is excellent. The new PC Emulator also means that over 95% of PC software (DBase, Lotus, Wordstar etc.) will run on the A3000 straight away.

RISC User

RISC User is the world's largest Archimedes Support Group, and publishes the leading magazine devoted exclusively to the A3000 and Archimedes.
An annual subscription is **£16.90 (UK)** and carries a multitude of benefits.
You may subscribe at the same time as placing an order.

Learning Curve Computers are in short supply - order yours now and we will supply on a first-come first-served basis.
BE ONE OF THE FIRST IN THE COUNTRY WITH THIS SUPERB SYSTEM.

ORDER FORM

Please supply:
- **Learning Curve Computer Only** (£803.85) ☐
- **Learning Curve Colour System** (£1033.80) ☐
- Information about the Learning Curve System ☐
- A RISC User Subscription (£16.90 UK) ☐

I enclose a cheque for £_____ *(Please add £8 for courier delivery)*
Or: Please debit my Access/Visa/Connect Card No____/____/____/____/____/
Card Expiry Date ____/____/ Signature _____

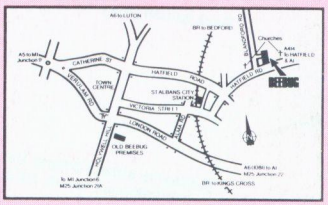
Name _____ RISC User/BEEBUG Memb No:_____
Address _____
_____ Post Code _____

HOW TO FIND BEEBUG

BY CAR
St Albans is easily reached from A1, A5, A6, M1 and M25

BY TRAIN
We are 10 minutes walk from St Albans City Station on the King's Cross to Bedford line.

Office Hours 9am - 5pm Mon - Fri
Showroom Hours 9am - 5:30pm Mon - Sat
Showroom is open until 8pm on Thursdays.



If you would like a leaflet on the Learning Curve Computer System write to:
BEEBUG Ltd 117 Hatfield Road, St Albans, Herts AL1 4JS. Phone (0727) 40303 Fax (0727) 60263